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—suited for mildly nonlinear data—is termed the metric-constrained union-of-subspaces (MC-UoS) model, which straddles the two extremes of the subspace model and the union-of-subspaces model. The second one of these models—suited for highly nonlinear data—is termed the metric-constrained kernel union-of-subspaces (MC-KUoS) model, which generalizes the kernel subspace 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

This work is supported in part 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’14) [1], and IEEE Workshop on Statistical Signal Processing (SSP’14) [2].

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).
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.

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 [3]–[7], denoising [8], [9], classification [10]–[13], and motion segmentation [14], [15]. These geometric models broadly fall into two categories, namely, linear models [3], [11], [16] and nonlinear models [4], [15], [17]–[19]. A further distinction can be made within each of these two categories depending upon whether the models are prespecified [20], [21] or learned from the data themselves [8], [15], [18], [22]–[24]. Our focus in this paper is on the latter case, since data-driven learning of geometric models is known to outperform prespecified geometric models [8], [25].

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) [3], [26], Karhunen–Loève transform [27], factor analysis [16], etc. But real-world data in many applications tend to be either mildly or highly 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 of these generalizations is the nonlinear manifold model [4], [7], [28], [29]. The 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 lie near a low-dimensional subspace [30]. Data-driven learning of geometric models in this case is commonly studied under the moniker of kernel PCA (KPCA) [28]. Another one of the most popular generalizations of linear models is the union-of-subspaces (UoS) model, which dictates that data lie near a mixture of low-dimensional subspaces in the ambient Hilbert space. Data-driven learning of the UoS model is commonly carried out under the rubrics of generalized PCA [31], hybrid linear modeling [32], dictionary learning [18], [33], subspace clustering [15], [34]–[37], mixture of factor analyzers [38], 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], [13], [15], [39]. 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 on the subspaces also 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 highly nonlinear data [28]. On the other hand, while the kernel subspace model can handle high nonlinearities in the data, one can reason that a single kernel subspace might require a large dimension to capture the richness of the entire data.

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. We are in particular interested in learning models for nonlinear data, which are either
mildly or highly nonlinear. One of our key objectives 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 in the paper is introduction of a novel geometric model, termed metric-constrained union-of-subspaces (MC-UoS) model, for mildly nonlinear data. 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 data mapped to some higher-dimensional Hilbert space (also known as the feature space) lie 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 \[17\], 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 in the presence of noise and missing dimensions, followed by denoising of (test) data using the learned geometry. (Other applications of our models will be investigated in future work.) 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 \[15\], \[24\], \[28\], \[33\], \[35\], \[37\], \[40\].

We conclude this discussion by pointing out that our work in this paper is not only related to the traditional literature on geometry learning, but it also has connections to the literature on clustering \[15\], \[35\], \[37\], \[40\]. Specifically, the mixture nature of 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 lower-case and upper-case letters for vectors and matrices, respectively. The \(i\)-th element of a vector \(v\) is denoted by \(v_{(i)}\), while \(a_{i,j}\) denotes the \((i, j)\)-th element of a matrix \(A\). The \(m\)-dimensional
zero vector is denoted by $\mathbf{0}_m$ and the $m \times m$ identity matrix is denoted by $I_m$. Given a set $\Omega$, $[A]_{\Omega}$ (resp., $[v]_{\Omega}$) denotes the submatrix of $A$ (resp., subvector of $v$) corresponding to the rows of $A$ (resp., entries of $v$) indexed by $\Omega$. Given two sets $\Omega_1$ and $\Omega_2$, $[A]_{\Omega_1,\Omega_2}$ denotes the submatrix of $A$ corresponding to rows and columns indexed by $\Omega_1$ and $\Omega_2$, respectively. Finally, $(\cdot)^T$ and $\text{tr}(\cdot)$ denote transpose and trace operations, respectively, while the Frobenius norm of a matrix $A$ is denoted by $\|A\|_F$ and the $\ell_2$ norm of a vector $v$ is represented by $\|v\|_2$.

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 \cite{7, 41}: $M = \bigcup_{\ell=1}^{L} S_\ell$, where $S_\ell$ is a subspace of $\mathbb{R}^m$. In here, we make the simplified assumption that all subspaces in $M$ have the same dimension, i.e., $\forall \ell$, $\dim(S_\ell) = s \ll m$. In this case, each subspace $S_\ell$ corresponds to a point on the Grassmann manifold $G_{m,s}$, which denotes the set of all $s$-dimensional subspaces of $\mathbb{R}^m$. While the canonical UoS model allows $S_\ell$’s to be arbitrary points on $G_{m,s}$, 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 $G_{m,s}$ and define an MC-UoS according to that metric as follows.

**Definition 1. (Metric-Constrained Union-of-Subspaces.)** A UoS $M = \bigcup_{\ell=1}^{L} S_\ell$ is said to be constrained with respect to a metric $d_u : G_{m,s} \times G_{m,s} \rightarrow [0, \infty)$ if $\max_{\ell, p, \ell \neq p} d_u(S_\ell, S_p) \leq \epsilon$ 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 \cite{42}. Specifically, if $D_\ell \in \mathbb{R}^{m \times s}$ and $D_p \in \mathbb{R}^{m \times s}$ denote orthonormal bases of subspaces $S_\ell$ and $S_p$, respectively, then

$$d_u(S_\ell, S_p) = \sqrt{s - \text{tr}(D_\ell^T D_p D_p^T D_\ell)} = \|D_\ell - P_{S_p} D_\ell\|_F,$$

We are using the terms “mildly nonlinear” and “highly nonlinear” here in an informal way. Heuristically, data that cannot be represented through a mixture of linear components should be deemed “highly nonlinear.”
where $P_{S_p}$ denotes the projection operator onto the subspace $S_p$: $P_{S_p} = D_pD_p^T$. 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 $G_{m,s}$ in [43]. Note that $d_u(\cdot, \cdot)$ in (1) is directly related to the concept of principal angles between two subspaces. Given two subspaces $S_t, S_p$ and their orthonormal bases $D_t, D_p$, the cosines of the principle angles $\cos(\theta_{t,p}^j), j = 1, \ldots, s$, between $S_t$ and $S_p$ are defined as the ordered singular values of $D_t^T D_p$ [35]. It therefore follows that $d_u(S_t, S_p) = \sqrt{s - \sum_{j=1}^s \cos^2(\theta_{t,p}^j)}$. 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 $\cos(\theta_{t,p}^j)$’s [44]. 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 drawn from an MC-UoS $\mathcal{M}$ 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 $N$ (likely noisy) training samples, $Y = [y_1, \ldots, y_N] \in \mathbb{R}^{m \times N}$, that are drawn from an MC-UoS $\mathcal{M} \subset \mathbb{R}^m$. Our goal is to learn $\mathcal{M}$ using the training data $Y$, 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 $\mathcal{M}$ in terms of the following optimization program:

$$\{S_\ell\}_{\ell=1}^L = \arg\min_{\{S_\ell\} \subset G_{m,s}} \sum_{\ell=1}^L d_u^2(S_\ell, S_p) + \lambda \sum_{i=1}^N \|y_i - P_{S_\ell} y_i\|_2^2,$$  

(2)

where $l_i = \arg\min_{\ell} \|y_i - P_{S_\ell} y_i\|_2^2$ with $P_{S_\ell} y_i$ denoting the (orthogonal) projection of $y_i$ onto the subspace $S_\ell$. 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 [45].

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 $Y$ 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 $Y$ 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 on synthetic and real data, we will focus on the metric of average approximation error of test data using the learned subspaces. 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 distance between the learned and the true subspaces.
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 $\mathbb{R}^m$ that, when mapped to a higher-dimensional feature space $\mathcal{F} \subset \mathbb{R}^{\tilde{m}}$ with $\tilde{m} \gg m$, can be modeled as an MC-UoS $\mathcal{M}$ in the feature space. Specifically, let $\phi : \mathbb{R}^m \to \mathcal{F} \subset \mathbb{R}^{\tilde{m}}$ be a nonlinear map from $\mathbb{R}^m$ to $\mathcal{F}$. We once again assume access to a collection of $N$ (likely noisy) training samples, $Y = [y_1, \ldots, y_N] \in \mathbb{R}^m \times N$, with the fundamental difference here being that the mapped training data $\phi(Y) = [\phi(y_1), \ldots, \phi(y_N)]$ is now assumed to be drawn from an MC-UoS $\mathcal{M} = \bigcup_{\ell=1}^L S_\ell \subset \mathbb{R}^{\tilde{m}}$, where $\ell \in \mathbb{R}$. Here, we also make the simplified assumption that $\text{rank}(\phi(Y)) = N$, which is justified as long as $\tilde{m} \gg N$ and no two training samples are identical. Our goal in this setting is to learn the (feature space) MC-UoS $\mathcal{M}$ using the training data $Y$, which in theory can still be achieved by solving the following variant of the optimization program (2):

$$\{S_\ell\}_{\ell=1}^L = \arg\min_{\{S_\ell\} \subset \mathbb{R}^{\tilde{m}}, \ell \neq \ell'} \sum_{\ell} d_\epsilon^2(S_\ell, S_{\ell'}) + \lambda \sum_{i=1}^N \left\| \phi(y_i) - P_{S_\ell} \phi(y_i) \right\|_2^2,$$

(3)

where $l_i = \arg\min_{l} \|\phi(y_i) - P_{S_l} \phi(y_i)\|_2^2$ with $P_{S_l} \phi(y_i)$ denoting the (orthogonal) projection of $\phi(y_i)$ onto the $s$-dimensional subspace $S_l$ in $\mathbb{R}^{\tilde{m}}$.

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” [17], 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 $\kappa(y, y') = \langle \phi(y), \phi(y') \rangle$ for all $y, y' \in \mathbb{R}^m$, 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 $Y$ are observed, while the second scenario corresponds to the case of “missing data” in which some dimensions of each training sample in $Y$ remain unobserved. Finally, we will quantify the performance of our MC-KUoS learning algorithms using the metric of average approximation error of test data. 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], [46], which will also be addressed in 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 the case of complete training data $Y$, we begin with centering of the data. This involves defining the mean of the samples in $Y$ as $\tilde{y} = \frac{1}{N} \sum_{i=1}^{N} y_i$ and then subtracting this mean from $Y$ to obtain the centered data $\tilde{Y} = [\tilde{y}_1, \ldots, \tilde{y}_N]$, where $\tilde{y}_i = y_i - \tilde{y}$, $i = 1, \ldots, N$. Next, we focus on simplification of the optimization problem \[2\]. To this end, we first define an $L \times N$ indicator matrix $W$ that identifies memberships of the $\tilde{y}_i$’s in different subspaces, where $w_{\ell, i} = 1$, $\ell = 1, \ldots, L$, $i = 1, \ldots, N$, if and only if $\tilde{y}_i \in S_\ell$. Mathematically,

$$W = [w_{\ell, i} \in \{0, 1\} : \forall i = 1, \ldots, N, \sum_{\ell=1}^{L} w_{\ell, i} = 1].$$  

(4)

Further, notice that $\|y_i - P_{S_\ell} y_i\|_2^2$ in \[2\] can be rewritten as

$$\|y_i - P_{S_\ell} y_i\|_2^2 = \|\tilde{y}_i - P_{S_\ell} \tilde{y}_i\|_2^2 = \|\tilde{y}_i\|_2^2 - \|D_\ell^T \tilde{y}_i\|_2^2,$$

(5)

where $D_\ell \in \mathbb{R}^{m \times s}$ denotes an (arbitrary) orthonormal basis of $S_\ell$. Therefore, defining $D = [D_1, \ldots, D_L]$ to be a collection of orthonormal bases of $S_\ell$’s, we can rewrite \[2\] as $(D, W) = \arg \min_{D, W} F_1(D, W)$ with the objective function $F_1(D, W)$ given by\[7\]

$$F_1(D, W) = \sum_{\ell, p=1}^{L} \| D_\ell - P_{S_p} D_\ell \|_F^2 + \lambda \sum_{i=1}^{N} \sum_{\ell=1}^{L} w_{\ell, i} (\|\tilde{y}_i\|_2^2 - \|D_\ell^T \tilde{y}_i\|_2^2).$$  

(6)

Minimizing \[6\] simultaneously over $D$ and $W$ is challenging and is likely to be computationally infeasible. Instead, we adopt an alternate minimization approach \[47\], \[48\], which involves iteratively solving \[6\] by alternating between the following two steps: (i) minimizing $F_1(D, W)$ over $W$ for a fixed $D$, which we term as the subspace assignment step; and (ii) minimizing $F_1(D, W)$ over $D$ for a fixed $W$, which we term as the subspace update stage. To begin this alternate minimization, we start with an initial $D$ in which each block $D_\ell \in \mathbb{R}^{m \times s}$ is a random orthonormal basis. Next, we fix this $D$ and carry out subspace assignment, which now amounts to solving

$$\forall i = 1, \ldots, N, \ l_i = \arg \min_{\ell = 1, \ldots, L} \|\tilde{y}_i - P_{S_\ell} \tilde{y}_i\|_2^2 = \arg \max_{\ell = 1, \ldots, L} \|D_\ell^T \tilde{y}_i\|_2^2$$

(7)

and then setting $w_{\ell, i} = 1$ and $w_{\ell', i} = 0 \ \forall \ell \neq \ell_i$. In order to move to the subspace update step, we fix the matrix $W$ and focus on optimizing $F_1(D, W)$ over $D$. However, this step requires more attention since minimizing over the entire $D$ at once will also lead to a large-scale optimization problem. We address this problem by once again resorting to block-coordinate descent (BCD) \[48\] and updating only one $D_\ell$ at a time while keeping the other $D_p$’s ($p \neq \ell$) fixed in \[6\]. In this regard, suppose we are in the process of updating $D_\ell$ for a fixed $\ell$ during the subspace update step. Define $c_\ell = \{i \in \{1, \ldots, N\} : w_{\ell, i} = 1\}$ to be the set containing the indices of all $\tilde{y}_i$’s that are assigned to $S_\ell$ (equivalently, $D_\ell$) and let $\tilde{Y}_\ell = [\tilde{y}_i : i \in c_\ell]$ be the corresponding $m \times |c_\ell|$ matrix. Then it can be shown after some manipulations of \[6\] that updating $D_\ell$ is equivalent to solving the following problem:

$$D_\ell = \arg \min_{Q \in \mathcal{V}_{m, s}} \sum_{p \neq \ell} \|Q - P_{S_p} Q\|_F^2 + \frac{\lambda}{2} (\|\tilde{Y}_\ell\|_F^2 - \|Q^T \tilde{Y}_\ell\|_F^2) = \arg \max_{Q \in \mathcal{V}_{m, s}} \text{tr} \left( Q^T (\sum_{p \neq \ell} D_p D_p^T + \frac{\lambda}{2} \tilde{Y}_\ell \tilde{Y}_\ell^T) Q \right),$$  

(8)

\[2\]Note that the minimization here is being carried out under the assumption of $D_\ell$’s being orthonormal and $W$ being described by \[4\].
Algorithm 1: Metric-Constrained Union-of-Subspaces Learning (MiCUSaL)

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

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

1: $\bar{y} \leftarrow \frac{1}{N} \sum_{i=1}^{N} y_i$, $\bar{y}_i \leftarrow y_i - \bar{y}$, $i = 1, \ldots, N$.
2: while stopping rule do
3:   for $i = 1$ to $N$ (Subspace Assignment) do
4:     $l_i \leftarrow \arg \max_{\ell} \|D_{\ell}^T \tilde{y}_i\|_2$.
5:     $w_{l,i} \leftarrow 1$ and $\forall \ell \neq l_i, w_{\ell,i} \leftarrow 0$.
6:   end for
7:   for $\ell = 1$ to $L$ (Subspace Update) do
8:     $c_{\ell} \leftarrow \{1 \leq i \leq N : w_{\ell,i} = 1\}$, $\tilde{Y}_{\ell} \leftarrow [\tilde{y}_i : i \in c_{\ell}]$.
9:     $A_{\ell} \leftarrow \sum_{p \neq \ell} D_p D_p^T + \frac{\lambda}{2} \tilde{Y}_{\ell} \tilde{Y}_{\ell}^T$.
10:    Eigen decomposition of $A_{\ell}$: $U_{\ell} \Sigma_{\ell} U_{\ell}^T = A_{\ell}$.
11:    $D_{\ell} \leftarrow$ Columns of $U_{\ell}$ corresponding to $s$-largest diagonal elements in $\Sigma_{\ell}$.
12: end for
13: end while

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

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. 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 $A_{\ell} = \sum_{p \neq \ell} D_p D_p^T + \frac{\lambda}{2} \tilde{Y}_{\ell} \tilde{Y}_{\ell}^T$. It then follows from [49] that (8) has a closed-form solution that involves eigen decomposition of $A_{\ell}$. Specifically, $D_{\ell} = \arg \max \text{tr}(D_{\ell}^T A_{\ell} D_{\ell})$ is given by the first $s$ eigenvectors of $A_{\ell}$ 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 learning. This algorithm, which we term metric-constrained union-of-subspaces learning (MiCUSaL), is given by Algorithm 1. 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 (8) 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).

### B. Practical Considerations of MC-UoS Learning Using Complete Data

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,
we first estimate the dimension of each subspace $S$ process before using the estimator. This involves first "denoising" our data by projecting provides a simple solution. However, the MLE of $[53]$ is sensitive to noise. We therefore first apply a "smoothing" $s$ for an incomplete list. In this paper, we focus on the method given in [53], which formulates the maximum $L$ have $\epsilon$ This process of finding the closest pair of subspaces and merging them is repeated until the normalized subspace $\text{Algorithm 1. By defining an } m$ $D$ removal of $\{S$ redundant subspaces from our collection of subspaces $\{S_\ell\}_{\ell=1}^{L_{max}}$ after each subspace assignment step. This involves removal of $D_\ell$ from $D$ if no signals in our training data get assigned to the subspace $S_\ell$. 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 subspaces that are too close to each other. In this step, we greedily merge pairs of closest subspaces as long as their normalized subspace $d_{\ell}\left(S_{\ell^*}, S_{p^*}\right)$ given by

$$ (\ell^*, p^*) = \arg \min_{\ell \neq p} d_u(S_\ell, S_p) \quad \text{s.t.} \quad d_u(S_{\ell^*}, S_{p^*}) \leq \epsilon_{min}. \quad (9) $$

We then merge $S_{\ell^*}$ and $S_{p^*}$ by setting $c_{\ell^*} \leftarrow c_{\ell^*} \cup c_{p^*}$ and $\bar{Y}_{\ell^*} \leftarrow \left[\hat{y}_i : i \in c_{\ell^*}\right]$, where $c_{\ell^*}, c_{p^*}$ are as defined in Algorithm 1. By defining an $m \times m$ symmetric matrix $A_{\ell^*} = \sum_{\ell \neq \ell^*, p} D_\ell D_\ell^T + \frac{1}{2} \bar{Y}_{\ell^*} \bar{Y}_{\ell^*}^T$, $D_\ell$ is then set equal to the first $s_{\max}$ eigenvectors of $A_{\ell^*}$ associated with its $s_{\max}$-largest eigenvalues. Finally, we remove $D_{p^*}$ from $D$.

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 $\epsilon_{\min}$. Without loss of generality, we assume to have $L$ subspaces left after this greedy subspace merging, where each $S_\ell$ ($\ell = 1, \ldots, L$) 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 $S_\ell$, 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., [50–53] for an incomplete list. In this paper, we focus on the method given in [53], which formulates the maximum likelihood estimator (MLE) of $s_\ell$. This is because: (i) the noise level is unknown in our problem, and (ii) [53] provides a simple solution. However, the MLE of [53] is sensitive to noise. We therefore first apply a “smoothing” process before using the estimator. This involves first “denoising” our data by projecting $\hat{Y}_\ell$ onto $S_\ell$, given by $\bar{Y}_\ell = D_\ell D_\ell^T \hat{Y}_\ell$, and then using $\hat{Y}_\ell$ to estimate $s_\ell$. By fixing a column $\hat{y}$ in $\hat{Y}_\ell$ and the number of nearest neighbors $k_0$, the unbiased MLE of $s_\ell$ with respect to $\hat{y}$ is given by [53]

$$ s_\ell^{k_0}(\hat{y}) = \left[ \frac{1}{k_0 - 2} \sum_{a=1}^{k_0-1} \log \frac{\Gamma_{k_0}(\hat{y})}{\Gamma_a(\hat{y})} \right]^{-1}, \quad (10) $$

December 23, 2014 DRAFT
\textbf{Algorithm 2:} Adaptive MC-UoS Learning (aMiCUSaL)

\textbf{Input:} Training data $Y \in \mathbb{R}^{m \times N}$, loose upper bounds $L_{\text{max}}$ and $s_{\text{max}}$, and parameters $\lambda$, $k_1$, $k_2$, $\epsilon_{\text{min}}$.

\textbf{Initialize:} Orthonormal bases \{$D_\ell \in \mathbb{R}^{m \times s_{\text{max}}}$\}, $L = L_{\text{max}}$.

1: $\bar{y} \leftarrow \frac{1}{N} \sum_{i=1}^{N} y_i$, $\bar{y}_i \leftarrow y_i - \bar{y}$, $i = 1, \ldots, N$.
2: \textbf{while} stopping rule \textbf{do}
3: \hspace{1em} Fix $D$ and update $W$ according to (7). Also, set $T \leftarrow \emptyset$ and $L_1 \leftarrow 0$.
4: \hspace{1em} \textbf{for} $\ell = 1$ to $L$ \textbf{do}
5: \hspace{2em} $c_\ell \leftarrow \{1 \leq i \leq N : w_{T,\ell,i} = 1\}$.
6: \hspace{2em} \textbf{if} $|c_\ell| \neq 0$ \textbf{then} $L_1 \leftarrow L_1 + 1$ \textbf{and} $T \leftarrow T \cup \{\ell\}$.
7: \hspace{1em} \textbf{end for}
8: \hspace{1em} \textbf{for} $\ell = 1$ to $L_1$ (Subspace Pruning) \textbf{do}
9: \hspace{2em} $c_\ell \leftarrow \{1 \leq i \leq N : w_{T,\ell,i} = 1\}$, $\bar{Y}_\ell \leftarrow [\bar{y}_i : i \in c_\ell]$.
10: \hspace{1em} \textbf{end for}
11: $D \leftarrow [D_{T,1}, \ldots, D_{T,L_1}]$ and $L \leftarrow L_1$.
12: Update each $D_\ell$ ($\ell = 1, \ldots, L$) in $D$ according to (8).
13: \textbf{end while}
14: $(\ell^*, p^*) = \arg \min_{\ell \neq p, \ell, p = 1, \ldots, L} d_u(S_{\ell}, S_p)$.
15: \textbf{while} $d_u(S_{\ell^*}, S_{p^*}) \leq \epsilon_{\text{min}}$ (Subspace Merging) \textbf{do}
16: \hspace{1em} Merge $S_{\ell^*}$ and $S_{p^*}$, and update $D_{\ell^*}$.
17: $D \leftarrow [D_1, \ldots, D_{\ell^* - 1}, D_{\ell^*}, D_{\ell^* + 1}, \ldots, D_{p^* - 1}, D_{p^* + 1}, \ldots, D_L]$ and $L \leftarrow L - 1$.
18: $(\ell^*, p^*) = \arg \min_{\ell \neq p, \ell, p = 1, \ldots, L} d_u(S_{\ell}, S_p)$.
19: \textbf{end while}
20: \textbf{for} $\ell = 1$ to $L$ \textbf{do}
21: \hspace{1em} $\bar{Y}_\ell \leftarrow [\bar{y}_i : i \in c_\ell]$ and $\hat{Y}_\ell \leftarrow D_\ell \bar{D}_\ell \bar{Y}_\ell$.
22: \hspace{1em} Estimate $s_\ell$ according to (10) and (11).
23: \hspace{1em} \textbf{end for}
24: $s \leftarrow \max\{s_1, \ldots, s_L\}$.
25: $\hat{D}_\ell \leftarrow$ First $s$ columns of $D_\ell$, $\ell = 1, \ldots, L$.
26: Initialize Algorithm 1 by \{$\hat{D}_\ell$\}_{\ell=1}^L and update \{$\hat{D}_\ell$\}_{\ell=1}^L by Algorithm 1.

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

where $\Gamma_a(y)$ is the $\ell_2$ distance from $\hat{y}$ to its $a$-th nearest neighbor in $\bar{Y}_\ell$. An estimate of $s_\ell$ can now be written as the average of all estimates with respect to every signal in $\bar{Y}_\ell$, i.e., $\hat{s}_\ell^{k_0} = \frac{1}{|c_\ell|} \sum_{i \in c_\ell} s_\ell^{k_0}(\hat{y}_i)$. In fact, as suggested
in [53], we calculate $s_{\ell}$ by averaging over a range of $k_0 = k_1, \ldots, k_2$, i.e.,

$$s_{\ell} = \frac{1}{k_2 - k_1 + 1} \sum_{k_0=k_1}^{k_2} s_{k_0}.$$  

(11)

Once we get an estimate $s = \max_{\ell} s_{\ell}$, we trim each orthonormal basis by keeping the first $s$ columns of each (ordered) orthonormal basis $D_\ell$ only in our collection, which is denoted by $\hat{D}_\ell$. Given the bases $\{\hat{D}_\ell \in \mathbb{R}^{m \times s}\}_{\ell=1}^L$, we finally perform MiCUSaL again that is initialized using these $\hat{D}_\ell$’s until it converges. Combining all the steps mentioned above, 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 with missing entries. To be specific, for each $y_i$ in $Y$, we assume to only observe its entries at locations given by the set $\Omega_i \subset \{1, \ldots, m\}$ with $|\Omega_i| > s$, which is denoted by $[y_i]_{\Omega_i} \in \mathbb{R}^{[|\Omega_i|]}$. Since we do not have access to the complete data, it is impossible to compute the quantities $\|y_i - P_{S_i[y_i]_{\Omega_i}}\|_2^2$ in (2) explicitly. But, the authors in [54] have shown that $\|y_i - P_{S_i[y_i]_{\Omega_i}}\|_2^2$ is very close to $\frac{|\Omega_i|}{m} \|y_i - P_{S_i[y_i]_{\Omega_i}}\|_2^2$ with very high probability as long as $|\Omega_i|$ is slightly greater than $s$. Here, $P_{S_i[\cdot]}$ is defined as $P_{S_i[\cdot]} = [D_i][\Omega_i,\{D_i\}^T_{\Omega_i,\ldots},\{D_i\}^T_{\Omega_i,\ldots}]^{-1} [D_i]_{\Omega_i,\ldots}$. Motivated by this, we replace $\|y_i - P_{S_i[y_i]_{\Omega_i}}\|_2^2$ by $\frac{m}{|\Omega_i|} \|y_i[y_i]_{\Omega_i} - P_{S_{\Omega_i},[y_i]_{\Omega_i}}\|_2^2$ in (2) and reformulate the MC-UoS learning problem as $(D, W) = \arg \min_{D, W} F_2(D, W)$, where

$$F_2(D, W) = \sum_{\ell \neq p}^L \|D_\ell - P_{S_p}D_\ell\|_F^2 + \lambda \sum_{i=1}^{N} \sum_{\ell = 1}^L w_{\ell, i} \frac{m}{|\Omega_i|} \|y_i[y_i]_{\Omega_i} - P_{S_{\Omega_i},[y_i]_{\Omega_i}}\|_2^2.$$  

(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 $D$ such that each block $D_\ell \in \mathbb{R}^{m \times s}$ is a random orthonormal basis. Next, when $D$ is fixed, subspace assignment corresponds to solving

$$\forall i = 1, \ldots, N, l_i = \arg \min_{\ell = 1, \ldots, L} \|y_i[y_i]_{\Omega_i} - P_{S_{\Omega_i},[y_i]_{\Omega_i}}\|_2^2.$$  

(13)

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

$$f_2(D_\ell) = -\text{tr}(D_\ell^T A_\ell D_\ell) + \frac{\lambda}{2} \sum_{i \in c_\ell} \frac{m}{|\Omega_i|} \|y_i[y_i]_{\Omega_i} - P_{S_{\Omega_i},[y_i]_{\Omega_i}}\|_2^2.$$  

Here, $c_\ell$ is as defined in Sec. III-A and $A_\ell = \sum_{p \neq \ell} D_pD_p^T$. It is also easy to verify that $f_2(D_\ell)$ is invariant to the choice of the orthonormal basis of $S_\ell$, hence we can treat $\min_{D_\ell \in \mathbb{R}^{m \times s}} f_2(D_\ell)$ as an optimization problem on the Grassmann manifold [55]. Note that we can rewrite $f_2(D_\ell)$ as $f_2(D_\ell) = \sum_{q=0}^{|c_\ell|} f_2^{(q)}(D_\ell)$, where $f_2^{(0)}(D_\ell) = -\text{tr}(D_\ell^T A_\ell D_\ell)$ and $f_2^{(q)}(D_\ell) = \frac{\lambda}{2} \sum_{i \in c_\ell} \frac{m}{|\Omega_i|} \|y_{c_\ell(q)}[y_{c_\ell(q)}]_{\Omega_i} - P_{S_{\Omega_i(c_\ell(q))},[y_{c_\ell(q)}]_{\Omega_i}}\|_2^2$ for $q = 1, \ldots, |c_\ell|$. In here, $c_\ell(q)$ denotes the $q$-th element in $c_\ell$. In order to minimize $f_2(D_\ell)$, we employ incremental gradient descent procedure [56] on Grassmann manifold, which performs the update with respect to a single component in each step. To be
specific, we first compute the gradient of one cost function $f_2^{(q)}(D_t)$ in $f_2(D_t)$, and move along a short geodesic curve in the gradient direction. For instance, the gradient of $f_2^{(0)}(D_t)$ is

$$\nabla f_2^{(0)} = (I_m - D_tD_t^T)\frac{df_2^{(0)}}{dD_t} = -2(I_m - D_tD_t^T)A_tD_t.$$ 

Then the geodesic equation emanating from $D_t$ in the direction $-\nabla f_2^{(0)}$ with a step length $\eta$ is given by

$$D_t(\eta) = D_tV_t\cos(\Sigma\eta)V_t^T + U_t\sin(\Sigma\eta)V_t^T,$$

where $U_t\Sigma V_t^T$ is the SVD decomposition of $-\nabla f_2^{(0)}$. The update of $D_t$ with respect to $f_2^{(q)}(D_t)(q = 1, \ldots, |c_\ell|)$ can be performed as in the GROUSE algorithm but with a step size $\frac{\lambda_m}{2d_{|c_\ell|}}\eta$. In order for $f_2$ to converge, we change the step size after each iteration. We conclude this section by presenting our learning algorithm for missing data in Algorithm 3, termed robust MC-UoS learning (rMiCUSaL).

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

**Input:** Training data $\{(y_i)_{\Omega_i}\}_{i=1}^N$, number of subspaces $L$, dimension of subspaces $s$, and parameters $\lambda$ and $\eta$.

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

1: while stopping rule do
2: for $i = 1$ to $N$ (Subspace Assignment) do
3: $l_i \leftarrow \arg \min \|y_i\Omega_i - P_{S_{c_{l_i}}}y_i\Omega_i\|^2_2$.
4: $w_{l_i,i} \leftarrow 1$ and $\forall \ell \neq l_i, w_{\ell,i} \leftarrow 0$.
5: end for
6: for $\ell = 1$ to $L$ (Subspace Update) do
7: $c_\ell \leftarrow \{1 \leq i \leq N : w_{\ell,i} = 1\}$, $t \leftarrow 0$.
8: while stopping rule do
9: $t \leftarrow t + 1$, $\eta_t \leftarrow \frac{\eta}{t}$.
10: $A_\ell \leftarrow \sum_{p \neq \ell} D_pD_p^T$, $\Delta_\ell \leftarrow 2(I_m - D_\ell D_\ell^T)A_\ell D_\ell$.
11: $D_\ell \leftarrow D_\ell V_\ell \cos(\Sigma\eta)V_\ell^T + U_\ell\sin(\Sigma\eta)V_\ell^T$, where $U_\ell \Sigma V_\ell^T$ is the compact SVD of $\Delta_\ell$.
12: for $q = 1$ to $|c_\ell|$ do
13: $\theta \leftarrow \left((D_\ell)_R^{T\left(\left(I_{|c_\ell|^2}\right)\right)} : [D_\ell]\omega_{|c_\ell|}^{T\left(\left(I_{|c_\ell|^2}\right)\right)}\right)^{-1}\left((D_\ell)_R^{T\left(\left(I_{|c_\ell|^2}\right)\right)} : [y_{|c_\ell|}]\omega_{|c_\ell|}^{T\left(\left(I_{|c_\ell|^2}\right)\right)}\right)$, $\omega \leftarrow D_\ell \theta$.
14: $r \leftarrow 0_m$, $[r]\omega_{|c_\ell|}^{T\left(\left(I_{|c_\ell|^2}\right)\right)} \leftarrow [y_{|c_\ell|}]\omega_{|c_\ell|}^{T\left(\left(I_{|c_\ell|^2}\right)\right)} - [\omega]\omega_{|c_\ell|}^{T\left(\left(I_{|c_\ell|^2}\right)\right)}$.
15: $D_\ell \leftarrow D_\ell + \left((\cos(\mu\frac{\lambda_m}{|\omega|}) - 1)\frac{\omega}{|\omega|} + \sin(\mu\frac{\lambda_m}{|\omega|})\frac{r}{|r|}\right)\theta^T$, where $\mu = \|r\|_2\|\omega\|_2$.
16: end for
17: end while
18: end for
19: end while

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

December 23, 2014 DRAFT
IV. MC-KUoS LEARNING FOR HIGHLY NONLINEAR DATA

In this section, we present algorithms to solve the problem of MC-KUoS learning from \( Y \in \mathbb{R}^{m \times N} \) for highly nonlinear data. We first generalize the MiCUSaL algorithm using the kernel trick \[17\] to learn an MC-KUoS with complete data. To deal with the case of “missing data,” we propose “kernel function value estimators” to solve \[3\]. Finally, we will discuss the solution of finding the “pre-images” of data in the feature space based on 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 \( Y \) to be \( G = \phi(Y)^T \phi(Y) \in \mathbb{R}^{N \times N} \), with its individual entries \( g_{i,j} = \kappa(y_i, y_j) \) for a pre-specified kernel \( \kappa : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R} \). Under the assumption that \( \text{rank}(\phi(Y)) = N \), the matrix \( G \) is positive definite. Similar to Algorithm \[1\], we begin with centering the \( \phi \)-mapped data in the feature space \( F \) as a pre-processing stage \[\text{I}\]. We denote the mean of the \( \phi \)-mapped “images” of \( Y \) by \( \bar{\phi} = \frac{1}{N} \sum_{i=1}^{N} \phi(y_i) \) and write the \( N \) centered “mapped training data” as \( \bar{\phi}(Y) = [\bar{\phi}(y_1), \ldots, \bar{\phi}(y_N)] \), where \( \bar{\phi}(y_i) = \phi(y_i) - \bar{\phi}, i = 1, \ldots, N \). The centered kernel matrix \( \bar{G} = \bar{\phi}(Y)^T \bar{\phi}(Y) \) can be calculated from \( G \) by

\[
\bar{G} = G - H_N G - G H_N + H_N G H_N,
\]

where \( H_N \) is an \( N \times N \) matrix with all elements \( \frac{1}{N} \). Then for any \( y, y' \in \mathbb{R}^m \), we have \[46\]

\[
\bar{\kappa}(y, y') = \bar{\phi}(y)^T \bar{\phi}(y') = \kappa(y, y') - \frac{1}{N} \sum_{i=1}^{N} \kappa(y_i, y_i) - \frac{1}{N} \sum_{i=1}^{N} \kappa(y_i, y_i) + \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \kappa(y_i, y_j),
\]

where \( 1_N = [1, 1, \ldots, 1]^T \) is an \( N \)-dimensional vector and \( k_y = [\kappa(y, y_1), \ldots, \kappa(y, y_N)]^T \in \mathbb{R}^N \). To write the expression in \[3\] in terms of inner products, we again use \( W \) to denote the membership indicator matrix as \[4\], where \( w_{i,\ell} = 1, \ell = 1, \ldots, L, i = 1, \ldots, N \), if and only if \( \bar{\phi}(y_i) \in S_\ell \). Let \( D = [D_1, \ldots, D_L] \), where \( D_\ell \) is an (arbitrary) orthonormal basis of \( S_\ell \). Then for any \( i = 1, \ldots, N \), we have the following

\[
\| \phi(y_i) - P_{S_\ell} \phi(y_i) \|^2 = \| \bar{\phi}(y_i) - P_{S_\ell} \bar{\phi}(y_i) \|^2 = \| \bar{\phi}(y_i) \|^2 - \| D_\ell^T \bar{\phi}(y_i) \|^2.
\]

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

\[
F_3(D, W) = \sum_{\ell, p=1}^{L} \| D_\ell - P_{S_p} D_\ell \|^2_F + \lambda \sum_{i=1}^{N} \sum_{\ell=1}^{L} w_{i,\ell} (\| \bar{\phi}(y_i) \|^2 - \| D_\ell^T \bar{\phi}(y_i) \|^2). \tag{17}
\]

Before discussing our algorithm to solve \( (17) \) using the kernel trick, we further simplify the terms in \( (17) \). We again use the definition \( c_\ell = \{ i \in \{1, \ldots, N \} : w_{i,\ell} = 1 \} \) to be the set containing the indices of all \( \bar{\phi}(y_i) \)'s that are assigned to \( S_\ell \), and let \( Y_\ell = [y_i : i \in c_\ell] \) be the corresponding \( m \times |c_\ell| \) matrix. Then the centered data which are assigned to subspace \( S_\ell \) can be denoted by \( \bar{\phi}(Y_\ell) = [\bar{\phi}(y_i) : i \in c_\ell] \). Since \( S_\ell \) is spanned by the columns of \( \bar{\phi}(Y_\ell) \), we can write \( D_\ell = \bar{\phi}(Y_\ell) E_\ell \), where \( E_\ell \in \mathbb{R}^{N_\ell \times s} \) is some basis representation matrix with \( N_\ell = |c_\ell| \), such

\[3\]In here, we will not need to explicitly center the data in the feature space. Instead, we only need to evaluate the inner products between the centered data in \( F \) for MC-KUoS learning.
that \( D_\ell \) is an orthonormal matrix. Therefore, it is easy to verify that \( E_\ell \) has to satisfy \( E_\ell^T [\bar{G}]_{\ell,\ell} E_\ell = I_s \), where \([\bar{G}]_{\ell,\ell} = \bar{\phi}(Y_\ell)^T \bar{\phi}(Y_\ell)\) denotes the centered kernel matrix for subspace \( S_\ell \). For this reason, instead of using \( D_\ell \) explicitly for computation, it suffices to use \( c_\ell \) and \( E_\ell \) for MC-KuoS learning and all the computations involving \( D_\ell \) can be carried out using \( c_\ell \), \( E_\ell \) and the kernel trick. Now notice that for any \( i = 1, \ldots, N \), we can write

\[
\| \bar{\phi}(y_i) \|^2 - \| D_\ell^T \bar{\phi}(y_i) \|^2 = \bar{\kappa}(y_i, y_i) - \| E_\ell^T \bar{\phi}(Y_\ell)^T \bar{\phi}(y_i) \|^2. \tag{18}
\]

where \( \bar{\kappa}(y_i, y_i) = \kappa(y_i, y_i) - \frac{2}{N} \mathbf{1}_N^T k(y_i) + \frac{1}{N^2} \mathbf{1}_N^T G \mathbf{1}_N \). To compute \( \bar{\phi}(Y_\ell)^T \bar{\phi}(y_i) \), we define \( \phi(Y_\ell) = [\phi(y_i) : i \in c_\ell] \) and let \( \Psi_\ell(y_i) = [\kappa(y_{i(1)}, y_i), \ldots, \kappa(y_{i(N_s)}, y_i)]^T \in \mathbb{R}^{N_s} \) be a vector with elements given by the inner products between \( \phi(y_i) \) and columns of \( \phi(Y_\ell) \). Then \( \Psi_\ell(y_i) = \bar{\phi}(Y_\ell)^T \bar{\phi}(y_i) = \Psi_p(y_i) - \frac{1}{N} \mathbf{1}_N \mathbf{1}^T k_{y_i} - \frac{1}{N} [G]_{c_\ell : 1_N + \frac{1}{N^2} \mathbf{1}_N^T G \mathbf{1}_N}. \) Therefore, we can write

\[
\| \bar{\kappa}(y_i, y_i) - \| E_\ell^T \bar{\phi}(Y_\ell)^T \bar{\phi}(y_i) \|^2 \| = s - \text{tr}(D_\ell^T D_p D_p^T D_\ell) = s - \text{tr}\left( \bar{\phi}(Y_\ell) E_\ell^T \bar{\phi}(Y_\ell) E_p(\bar{\phi}(Y_\ell) E_p)^T \right)
\]

\[
= s - \text{tr}\left( E_\ell^T [\bar{G}]_{c_\ell, c_p} E_p^T [\bar{G}]_{c_p, c_\ell} E_\ell \right), \tag{19}
\]

where \([\bar{G}]_{c_\ell, c_p} = \bar{\phi}(Y_\ell)^T \bar{\phi}(y_i)\) denotes the centered inter-subspace kernel matrix between \( S_\ell \) and \( S_p \).

Now we can reason about our algorithm in detail. Similar to MiCUsaL, we minimize \( F_3(D, W) \) by alternating between (i) minimizing \( F_3(D, W) \) over \( W \) for a fixed \( D \) (the kernel subspace assignment step) and (ii) minimizing \( F_3(D, W) \) over \( D \) for a fixed \( W \) (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 \( D_\ell \) of \( S_\ell \) can be represented as \( D_\ell = \bar{\phi}(Y_\ell) E_\ell \) and we can compute \( E_\ell \) explicitly by using \([\bar{G}]_{\ell,\ell}\). Therefore the initialization of \( D_\ell \) can be treated as initializing \( c_\ell \). Note that any \( s \) linear independent vectors define an \( s \)-dimensional subspace. Therefore, to initialize \( c_\ell \), we only 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. Since we assume that all \( \phi(y_i) \)’s are linearly independent, we can randomly choose \( s \) indexes from \( \mathcal{I}_N = \{1, \ldots, N\} \) without replacement to accomplish this goal. We list our initialization method in Algorithm 4 referred to as kernel initial-orthogonalization procedure (KIOP). Note that \( \bigcap_{\ell=1}^L c_\ell \neq \emptyset \) and we compute \( E_\ell \) by \( E_\ell = U_\ell \Sigma_\ell^{-\frac{1}{2}} \) in Algorithm 4. Since \( D_\ell = \bar{\phi}(Y_\ell) E_\ell \), it is
easy to convince oneself that $D^T \kappa D = I_s$ in this case.

We now move onto the kernel subspace assignment step. When $D$ (equivalently, $c_\ell$’s and $E_\ell$’s) is fixed, kernel subspace assignment corresponds to first solving
\[\forall i = 1, \ldots, N, \quad l_i = \arg \min_{\ell = 1, \ldots, L} \|\tilde{\phi}(y_i) - P_{S_{l_i}} \tilde{\phi}(y_i)\|_2^2 = \arg \min_{\ell = 1, \ldots, L} \|\tilde{\phi}(y_i, y_i) - \|E_T^T \tilde{\psi}_\ell(y_i)\|_2^2 \quad (20)\]
and then setting $w_{l_i,i} = 1$ and $w_{l,i} = 0 \forall \ell \neq l_i$. Next, for the kernel subspace update stage, since $W$ is fixed, all the $c_\ell$’s and $Y_\ell$’s are fixed. By writing $D_\ell = \tilde{\phi}(Y_\ell)E_\ell$, the problem of (17) for a fixed $W$ can be written as a

---

**Algorithm 5: Metric-Constrained Kernel UoS Learning (MC-KUSaL)**

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

1. Compute kernel matrix $G$ such that $g_{i,j} = \kappa(y_i, y_j)$.
2. $G \leftarrow G - H_N G - G H_N + H_N G H_N$.
3. Initialize $\{c_\ell\}_{\ell=1}^L$ and $\{E_\ell\}_{\ell=1}^L$ using KIOP (Algorithm 4).
4. **while** stopping rule **do**
   5. **for** $i = 1$ to $N$ (Kernel Subspace Assignment) **do**
      6. $l_i \leftarrow \arg \min_{\ell = 1, \ldots, L} \|E_T^T \tilde{\psi}_\ell(y_i)\|_2^2$.
      7. $w_{l_i,i} \leftarrow 1$ and $\forall \ell \neq l_i, w_{l,i} \leftarrow 0$.
   8. **end for**
9. **for** $\ell = 1$ to $L$ (Kernel Bases Initialization) **do**
   10. $c_\ell \leftarrow \{1 \leq i \leq N : w_{l,i} = 1\}$ and $N_\ell \leftarrow |c_\ell|$.
   11. Eigen decomposition of $[G]_{c_\ell,c_\ell} = U_\ell \Sigma_\ell U_\ell^T$, with the diagonal elements of $\Sigma_\ell$ in nonincreasing order.
   12. $E_\ell \leftarrow [U_{\ell:1,:}, \Sigma_\ell^{\frac{1}{2}}, I_{N_\ell:L,:}]$.
13. **end for**
14. **while** stopping rule **do**
   15. **for** $\ell = 1$ to $L$ (Kernel Subspace Update) **do**
      16. $A_\ell \leftarrow \sum_{p \neq \ell} [G]_{c_\ell,c_p} E_T^p E_T^T [G]_{c_p,c_\ell} + \frac{1}{2} [G]^2_{c_\ell,c_\ell}$.
      17. $E_\ell \leftarrow$ Eigenvectors corresponding to $s$-largest eigenvalues for the generalized problem $A_\ell b = \zeta [G]_{c_\ell,c_\ell} b$ such that $E_\ell^T [G]_{c_\ell,c_\ell} E_\ell = I_s$.
   18. **end for**
19. **end while**
20. **end while**

**Output:** $\{N_\ell \in \mathbb{N}\}_{\ell=1}^L$, $\{c_\ell\}_{\ell=1}^L$ and $\{E_\ell \in \mathbb{R}^{N_\ell \times s}\}_{\ell=1}^L$. 
function of $E_\ell$’s as follows:

$$
\min_{E_1, \ldots, E_L} f_3(E_1, \ldots, E_L) = \sum_{\ell, p=1}^{L} \left( \frac{1}{2} \|\bar{\phi}(Y_\ell)\|_F^2 + \frac{\lambda}{2} \left( \|\bar{\phi}(Y_\ell)\|_F^2 - \|E_\ell^T \bar{\phi}(Y_\ell)^T \bar{\phi}(Y_\ell)\|_F^2 \right) \right)
$$

s.t. $E_\ell^T [\bar{G}]_{c_\ell, c_\ell} E_\ell = I_s, \ell = 1, 2, \ldots, L.$ \hfill (21)

Instead of updating all the $E_\ell$’s simultaneously, which is again a difficult optimization problem, we use BCD to minimize $f_3$ and update $E_\ell$’s sequentially. Unlike MC-UoS learning, however, we have to be careful here since the number of samples in $Y$ that belong to $Y_\ell$ (i.e., $N_\ell$) may change after each subspace assignment step. In particular, we first need to initialize all the $E_\ell$’s such that $E_\ell \in \mathbb{R}^{N_\ell \times s}$ and $E_\ell^T [\bar{G}]_{c_\ell, c_\ell} E_\ell = I_s$. To do so, we again apply eigen decomposition of $[\bar{G}]_{c_\ell, c_\ell}$ with the diagonal entries of $\Sigma_\ell$ in nonincreasing order. Then we define $\mathcal{I}_\ell = \{1, \ldots, s\}$ and $E_\ell = [U_\ell : \ldots : [\Sigma_\ell]_{\mathcal{I}_\ell}^\frac{1}{2}]$. After this bases initialization step, we are ready to update $E_\ell$’s sequentially and after some manipulations, each BCD subproblem of (21) can be expressed as

$$
E_\ell = \arg \min_{Q : Q^T = \frac{1}{2} \|\bar{\phi}(Y_\ell)\|_F^2 + \frac{\lambda}{2} \left( \|\bar{\phi}(Y_\ell)\|_F^2 - \|Q^T \bar{\phi}(Y_\ell)^T \bar{\phi}(Y_\ell)\|_F^2 \right) = \mathbf{0}} \sum_{p \neq \ell} \|\bar{\phi}(Y_\ell)Q - P_{E_p}(\bar{\phi}(Y_\ell)Q)\|_F^2 + \frac{\lambda}{2} \left( \|\bar{\phi}(Y_\ell)\|_F^2 - \|Q^T \bar{\phi}(Y_\ell)^T \bar{\phi}(Y_\ell)\|_F^2 \right)
$$

where $A_\ell = \sum_{p \neq \ell} [\bar{G}]_{c_\ell, c_\ell} E_p E_p^T [\bar{G}]_{c_\ell, c_\ell} + \frac{\lambda}{2} [\bar{G}]_{c_\ell, c_\ell}^2$ is a symmetric matrix of dimension $N_\ell \times N_\ell$. Note that (22) has a similar intuitive interpretation as (8). When $\lambda = 0$, (22) reduces to the problem of finding a subspace which is closest to the remaining $L - 1$ subspaces in the feature space. When $\lambda = \infty$, (22) reduces to the kernel PCA problem $[28]$. Since $[\bar{G}]_{c_\ell, c_\ell}$ is a positive definite matrix, it again follows from [49] that the trace of $E_\ell^T A_\ell E_\ell$ is maximized when $E_\ell = [b_1, \ldots, b_s]$ is the set of eigenvectors associated with the $s$-largest eigenvalues for the generalized problem $A_\ell b = \zeta [\bar{G}]_{c_\ell, c_\ell} b$, with $E_\ell^T [\bar{G}]_{c_\ell, c_\ell} E_\ell = I_s$. The whole algorithm can be detailed in Algorithm [5] which we refer to as metric-constrained kernel union-of-subspaces learning (MC-KUoS).

B. MC-KUoS Learning Using Missing Data

In this section, we focus on MC-KUoS learning for the case of training data with missing entries in the input space. Our setup is similar to the one in Sec. III-C. That is, for $i = 1, \ldots, N$, we observe $y_i$ only at locations $\Omega_i \subset \{1, \ldots, m\}$. In the following, the resulting observed vector of $y_i$ is denoted by $[y_i]_{\Omega_i} \in \mathbb{R}^{[\Omega_i]}$. In order to support our theory, we will assume that indices of each observed signal, $\Omega_i$, of $y_i$ are drawn uniformly at random with replacement from $\{1, \ldots, m\}$. Note that the results derived in here can also be translated to the case of sampling $\Omega$ without replacement (we refer the reader to [58] Lemma 1] as an example). Given the missing data aspect of this problem 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 to use a robust estimate of the kernel function value $\kappa(y_i, y_j)$ using incomplete data $[y_i]_{\Omega_i}$ and $[y_j]_{\Omega_j}$. Mathematically, our goal is to find a proxy
function \( h(\cdot, \cdot) \) such that \( h([y_i]_{\Omega_i}, [y_j]_{\Omega_j}) \approx \kappa(y_i, y_j) \). To derive this proxy function, we start by considering the relationship between \([y_i]_{\Omega_i}, [y_j]_{\Omega_j}\) and \(y_i, y_j\) underlying different types of kernel functions.

We first consider isotropic kernels of the form \( \kappa(y_i, y_j) = k(||y_i - y_j||^2) \) for our analysis. To begin, we define \( z_{ij} = y_i - y_j \) and \( \Omega_{ij} = \Omega_i \cap \Omega_j \), resulting in \([z_{ij}]_{\Omega_{ij}} = [y_i]_{\Omega_i} - [y_j]_{\Omega_j} \in \mathbb{R}^{|\Omega_{ij}|}\). For any vector \( z_{ij} \), the authors in [54] have derived the coherence of a subspace spanned by a vector \( z_{ij} \) to be \( \mu(z_{ij}) = \frac{m||z_{ij}||^2}{||z_{ij}||^2} \) and shown that \( ||z_{ij}||_{\Omega_{ij}}||^2 \) is very close to \( \frac{m||z_{ij}||^2}{m} ||z_{ij}||^2 \) with high probability. Because of this result, we can give the following corollary that is essentially due to [54] Lemma 1 by plugging in the definition of \( z_{ij} \) and \([z_{ij}]_{\Omega_{ij}}\).

**Corollary 1.** Let \( \delta > 0, \Omega_{ij} = \Omega_i \cap \Omega_j \) and \( \alpha = \sqrt{\frac{2m||y_i - y_j||^2}{|\Omega_{ij}|}} \log(\frac{1}{\delta}) \). Then with probability at least \( 1 - 2\delta \),

\[
(1 - \alpha)||y_i - y_j||^2 \leq \frac{m}{|\Omega_{ij}|} ||[z_{ij}]_{\Omega_{ij}}||^2 \leq (1 + \alpha)||y_i - y_j||^2.
\]

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

**Theorem 1.** Let \( \delta > 0, \Omega_{ij} = \Omega_i \cap \Omega_j \) and \( \alpha = \sqrt{\frac{2m||y_i - y_j||^2}{|\Omega_{ij}|}} \log(\frac{1}{\delta}) \). Then for a Gaussian kernel \( \kappa(y_i, y_j) \), with probability at least \( 1 - 2\delta \), we have

\[
h([y_i]_{\Omega_i}, [y_j]_{\Omega_j}) \leq \kappa(y_i, y_j) \leq h([y_i]_{\Omega_i}, [y_j]_{\Omega_j})^{\frac{1}{1+\alpha}}.
\]

We skip the proof of this theorem since it is elementary. We should also note here that \( h([y_i]_{\Omega_i}, [y_j]_{\Omega_j}) = \kappa(y_i, y_i) = 1 \) as a special case for Gaussian kernels.

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

**Lemma 1.** Let \( \delta > 0, \Omega_{ij} = \Omega_i \cap \Omega_j \) and \( \beta = \sqrt{\frac{2m^2||y_i \circ y_j||^2}{|\Omega_{ij}|}} \log(\frac{1}{\delta}) \). Then with probability at least \( 1 - 2\delta \),

\[
\langle y_i, y_j \rangle - \beta \leq \frac{m}{|\Omega_{ij}|} \langle [y_i]_{\Omega_i}, [y_j]_{\Omega_j} \rangle \leq \langle y_i, y_j \rangle + \beta.
\]  

**Proof:** See Appendix A. 

The above lemma establishes that \( \langle [y_i]_{\Omega_i}, [y_j]_{\Omega_j} \rangle \) is close to \( \frac{m}{m} \langle y_i, y_j \rangle \) 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 \( \kappa(y_i, y_j) = (\langle y_i, y_j \rangle + c)^d \) with \( d > 0 \) and \( c \geq 0 \), we have \( h([y_i]_{\Omega_i}, [y_j]_{\Omega_j}) = (\frac{m}{|\Omega_{ij}|} \langle [y_i]_{\Omega_i}, [y_j]_{\Omega_j} \rangle + c)^d \).

To analyze the bounds on estimated kernel function value in this case, notice that if (23) holds and \( d \) is odd, we
will have the following
\[(\langle y_i, y_j \rangle - \beta + c)^d \leq \left( \frac{m}{|\Omega_{ij}|} \langle [y_i]_{\Omega_{ij}}, [y_j]_{\Omega_{ij}} \rangle + c \right)^d \leq (\langle y_i, y_j \rangle + \beta + c)^d.\]

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

**Theorem 2.** Let \(\delta > 0, \Omega_{ij} = \Omega_i \cap \Omega_j\) and \(\beta = \sqrt{\frac{2m\|y_i, y_j\|_2^2}{|\Omega_{ij}|}} \log(\frac{1}{\delta})\). Then for a polynomial kernel \(\kappa(y_i, y_j)\) with an odd degree \(d\), with probability at least \(1 - 2\delta\), we have
\[
(h([y_i]_{\Omega_j}, [y_j]_{\Omega_j})^{\frac{1}{d}} - \beta)^d \leq \kappa(y_i, y_j) \leq (h([y_i]_{\Omega_j}, [y_j]_{\Omega_j})^{\frac{1}{d}} + \beta)^d.
\]

Based on the discussion above, we can estimate the kernel function value \(\kappa(y_i, y_j)\) using the associated proxy function \(h(\cdot, \cdot)\) with entries of \(y_i\) and \(y_j\) belonging to \(\Omega_{ij}\) only. Therefore we can also compute an estimated kernel matrix \(G \in \mathbb{R}^{N \times N}\), defined as \(g_{i,j} = h([y_i]_{\Omega_j}, [y_j]_{\Omega_j})\) in the case of missing data. But the positive definiteness of \(G\) is not guaranteed. In this setting, we therefore first need to find a positive definite matrix \(\hat{G} \approx G\) before we can carry on with MC-KUoS learning. To deal with this issue, we begin with eigen decomposition of \(G = U\Lambda U^T\), where \(\Lambda = \text{diag}\{\lambda_{G}^{(1)}, \ldots, \lambda_{G}^{(N)}\}\) contains eigenvalues of \(G\). The resulting approximated kernel matrix \(\hat{G}\) that is “closest” to \(G\) can then be calculated by \(\hat{G} = U\hat{\Lambda}U^T\), where \(\hat{\Lambda} = \text{diag}\{\lambda_{\hat{G}}^{(1)}, \ldots, \lambda_{\hat{G}}^{(N)}\}\) and each \(\lambda_{\hat{G}}^{(i)}, i = 1, \ldots, N\), is defined as
\[
\lambda_{\hat{G}}^{(i)} = \begin{cases} 
\lambda_{G}^{(i)}, & \lambda_{G}^{(i)} > 0 \\
\delta_{\text{min}}, & \lambda_{G}^{(i)} = 0 \\
-\lambda_{G}^{(i)}, & \lambda_{G}^{(i)} < 0.
\end{cases}
\]

Here, \(\delta_{\text{min}} > 0\) is a predefined (arbitrarily small) parameter. Using the above procedure, one can obtain a positive definite matrix \(\hat{G}\) such that \(\hat{g}_{i,j} \approx \kappa(y_i, y_j)\) 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 \(\hat{g}_{i,j}\), obtained from \(h([y_i]_{\Omega_j}, [y_j]_{\Omega_j})\), in lieu of \(\kappa(y_i, y_j)\) in the overall learning process, which includes both kernel subspace assignment and kernel subspace update stages. We dub this approach as **robust MC-KUoS learning** (rMC-KUoS-L). We conclude this section by noting that we can also robustify classical kernel PCA by using \(\hat{G}\) as a means of learning kernel PCA with missing data, which we call rKPCA in our experiments.

**C. Pre-Image Reconstruction**

Thus far in this section, we have proposed the solutions of 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 \(z = x + \xi \in \mathbb{R}^m\), where \(\xi\) is a noise term and \(\tilde{\phi}(x) = \phi(x) - \phi\) belongs to one of the subspaces in \(\mathcal{M}\) (assume \(\tilde{\phi}(x) \in \mathcal{S}_\tau, \tau \in \{1, \ldots, L\}\)). In most information processing tasks, one needs to first find a representation of this sample \(z\) in terms of the learned MC-KUoS, which is akin to “denoising” \(z\). The “denoised sample” in the feature space is the projection of \(\phi(z)\) onto \(\mathcal{S}_\tau\), which is given by \(P_{\mathcal{S}_\tau}\phi(z) = D_\tau D_\tau^T \phi(z) + \tilde{\phi}\) with \(\tilde{\phi}(z) = \phi(z) - \tilde{\phi}\). However, in order to visualize the “denoised” sample in the ambient space, we more often than not need to project \(P_{\mathcal{S}_\tau}\phi(z)\) onto the input space in
many applications \cite{9, 59}, which is termed as pre-image reconstruction. In this section, we consider the problem of pre-image reconstruction based on MC-KUoS model to address this issue.

Mathematically, the problem of pre-image reconstruction can be stated as follows. We are given \( z \in \mathbb{R}^m \) and we are interested in finding \( \tilde{z} \in \mathbb{R}^m \) whose mapping to the feature space is closest to the projection of \( \phi(z) \) onto the learned MC-UoS in \( \mathcal{F} \). This involves first finding the index \( \tau \) such that \( \tau = \arg\min_{\ell} \|\phi(z) - P_{S,\ell} \phi(z)\|^2 \), which can be easily done using the kernel subspace assignment step described in \cite{20}. Next, we need to solve \( \tilde{z} = \arg\min_{\rho \in \mathbb{R}^m} \|\phi(\rho) - P_{S,\ell} \phi(z)\|^2 \). To solve this problem, we leverage the ideas in \cite{46, 60} that only use feature-space distances to find \( \tilde{z} \) (equivalently, to find pre-image of \( P_{S,\ell} \phi(z) \)). We first study this problem when the training samples in \( Y \) are complete.

1) Pre-Image Reconstruction Using Complete Data: We begin by first calculating the squared “feature distance” between the projection \( P_{S,\ell} \phi(z) \) and any \( \phi(y_i), i = 1, \ldots, N \), defined as \cite{46}

\[
d^2_F(\phi(y_i), P_{S,\ell} \phi(z)) = \|P_{S,\ell} \phi(z)\|^2 + \|\phi(y_i)\|^2 - 2(P_{S,\ell} \phi(z))^T \phi(y_i).
\]  

Notice that \( \|P_{S,\ell} \phi(z)\|^2 \) and \( (P_{S,\ell} \phi(z))^T \phi(y_i) \) can be calculated in terms of kernel representation as follows:

\[
\|P_{S,\ell} \phi(z)\|^2 = \tilde{\phi}(z)^T D_r D_r^T \phi(z) + \tilde{\phi}^T \tilde{\phi} + 2\tilde{\phi}(z)^T D_r D_r^T \phi = \tilde{\phi}(z)^T \tilde{\phi}(Y) E_r E_r^T \phi(Y)^T \phi(z) + \frac{1}{N^2} 1_N^T G 1_N + \frac{2}{N} \tilde{\phi}(z)^T \phi(Y) E_r E_r^T \phi(Y)^T \phi(Y) 1_N
\]

\[
= \tilde{\psi}_\tau(z)^T E_r E_r^T \left( \tilde{\psi}_\tau(z) (\frac{2}{N} N_{e,\tau} 1_N - \frac{2}{N^2} 1_N^T G 1_N) + \frac{1}{N^2} 1_N^T G 1_N \right)
\]

and

\[
(P_{S,\ell} \phi(z))^T \phi(y_i) = \tilde{\psi}_\tau(z)^T E_r E_r^T \left( \psi_\tau(y_i) (\frac{1}{N} 1_N^T k_{y_i}) + \frac{1}{N} 1_N^T k_{y_i} \right)
\]

Therefore, (24) becomes

\[
d^2_F(\phi(y_i), P_{S,\ell} \phi(z)) = \tilde{\psi}_\tau(z)^T E_r E_r^T \left( \psi_\tau(z) (\frac{1}{N} 1_N^T k_{y_i}) + \frac{1}{N} 1_N^T k_{y_i} \right)
\]

with \( g_{\tau,i} = \kappa(y_i, y_i) \).

We now describe our method for pre-image reconstruction using the Gaussian kernel first. In this case, the problem of minimizing \( \|\phi(\rho) - P_{S,\ell} \phi(z)\|^2 \) is equivalent to maximizing the function \( \rho(\tilde{z}) = (P_{S,\ell} \phi(z))^T \phi(\tilde{z}) \) \cite{9}, whose extremum can be obtained by setting \( \nabla_{\tilde{z}} \rho = 0 \), where \( \nabla_{\tilde{z}} \rho \) denotes the gradient of \( \rho \) with respect to \( \tilde{z} \). To do so, we express \( \rho(\tilde{z}) \) as

\[
\rho(\tilde{z}) = (D_r D_r^T \phi(z) + \tilde{\phi})^T \phi(z) = \tilde{\phi}(z)^T \tilde{\phi}(Y) E_r E_r^T \phi(Y)^T \phi(z) + \frac{1}{N} 1_N^T \phi(Y)^T \phi(z)
\]

\[
= \psi_\tau(z)^T E_r E_r^T (\psi_\tau(z) (\frac{1}{N} 1_N^T k_{\tilde{z}}) + \frac{1}{N} 1_N^T k_{\tilde{z}})
\]

\[
= \zeta_\tau(z)^T (\psi_\tau(z) (\frac{1}{N} 1_N^T k_{\tilde{z}}) + \frac{1}{N} 1_N^T k_{\tilde{z}}),
\]  

where \( \zeta_\tau(z) = E_r E_r^T \psi_\tau(z) \in \mathbb{R}^{N!} \). Next, we define \( \gamma = \frac{1}{N} (1 - \zeta_\tau(z) 1_N^T) 1_N \in \mathbb{R}^N \) and let \( \tilde{\gamma} \) be an \( N \)-dimensional vector such that \( [\tilde{\gamma}]_e = [\gamma]_e, + \zeta_\tau(z) \) and \( [\tilde{\gamma}]_{I_N \setminus e} = [\gamma]_{I_N \setminus e} \) (recall that \( I_N = \{1, \ldots, N\} \) and \( c_\tau \)
contains all the indexes of $\tilde{\phi}(y_i)$’s that are assigned to $S_r$, which means $\rho(\tilde{z}) = \gamma^T k_{\tilde{z}} = \sum_{i=1}^{N} \tilde{\gamma}^{(i)} \kappa(\tilde{z}, y_i)$. By setting $\nabla_{\tilde{z}} \rho = 0$, we get

$$\tilde{z} = \frac{\sum_{i=1}^{N} \tilde{\gamma}^{(i)} \exp(-\|\tilde{z} - y_i\|^2/2c)}{\sum_{i=1}^{N} \tilde{\gamma}^{(i)} \exp(-\|\tilde{z} - y_i\|^2/2c)}.$$

By using the approximation $P_{S_r} \phi(z) \approx \phi(\tilde{z})$ and the relation $\|\tilde{z} - y_i\|^2 = -c \log(\frac{1}{2} (2 - d_{z}^2(\phi(y_i), \phi(3))))$ [46], a unique pre-image can now be obtained by the following formula:

$$\tilde{z} = \frac{\sum_{i=1}^{N} \tilde{\gamma}^{(i)} \left( \frac{1}{2} (2 - d_{z}^2(P_{S_r} \phi(z), \phi(y_i))) \right) y_i}{\sum_{i=1}^{N} \tilde{\gamma}^{(i)} \left( \frac{1}{2} (2 - d_{z}^2(P_{S_r} \phi(z), \phi(y_i))) \right)}.$$

(26)

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

$$\tilde{z} = \frac{\sum_{i=1}^{N} \tilde{\gamma}^{(i)} \left( (P_{S_r} \phi(z))^T \phi(y_i) \right) \|P_{S_r} \phi(z)\|^2_{d}}{\sum_{i=1}^{N} \tilde{\gamma}^{(i)} \left( \frac{1}{2} (2 - d_{z}^2(P_{S_r} \phi(z), \phi(y_i))) \right)}.$$

(27)

2) Pre-Image Reconstruction Using Missing Data: We next consider the problem of reconstructing the pre-image of $P_{S_r} \phi(z)$ when the training samples have missing entries. As can be easily seen from [26], the solution of a pre-image for the Gaussian kernel can be written as $\tilde{z} = \frac{\sum_{i=1}^{N} e_i y_i}{\sum_{i=1}^{N} e_i}$, where $e_i = \tilde{\gamma}^{(i)} \left( \frac{1}{2} (2 - d_{z}^2(P_{S_r} \phi(z), \phi(y_i))) \right)$.

Similarly, from [27], we can also write the solution of $\tilde{z}$ to be $\tilde{z} = \frac{\sum_{i=1}^{N} e_i y_i}{\sum_{i=1}^{N} e_i}$ for the polynomial kernel, where $e_i = \tilde{\gamma}^{(i)} \left( \frac{(P_{S_r} \phi(z))^T \phi(y_i)}{\|P_{S_r} \phi(z)\|^2_{d}} \right)$ 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 $\kappa(z, y_i)$ using entries of $z$ belonging to $\Omega_i$ and $[y_i]_{\Omega_i}$, where the estimated kernel function value is denoted by $h(z, [y_i]_{\Omega_i})$.

Based on the estimated kernel function values $h(z, [y_i]_{\Omega_i})$’s, we can then find the solution of $\tau$ such that $\tau = \arg \min_{\tau} \|\tilde{\phi}(z) - P_{S_r} \phi(z)\|^2_{2}$, 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 $\tilde{z}$ separately in this case. To be specific, for the $u$-th entry of $\tilde{z}$, $u = 1, \ldots, m$, we define $r_u$ to be the set containing the indexes of the samples $[y_i]_{\Omega_i}$’s whose $u$-th entry are observed. Then $\tilde{z}(u) = \frac{\sum_{i \in r_u} e_i y_i(u)}{\sum_{i \in r_u} e_i}$ for the Gaussian kernel and $\tilde{z}(u) = \sum_{i \in r_u} e_i y_i(u)$ 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 $z$ 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/MC-KUoS from the complete/missing noisy training data, followed by denoising of complete noisy test samples using the learned geometric structures.

A. Experiments for MC-UoS Learning

In this section, we examine the effectiveness of MC-UoS learning using Algorithms [13]. For the complete data experiments, we compare MiCUSaL/aMiCUSaL with several state-of-the-art subspace learning algorithms such
as Block-Sparse Dictionary Design (SAC+BK-SVD) \[33\], \(K\)-subspace clustering (\(K\)-sub) \[40\], Sparse Subspace Clustering (SSC) \[15\], Robust Sparse Subspace Clustering (RSSC) \[35\], Thresholding-based Subspace Clustering (TSC) \[37\] and Principal Component Analysis (PCA) \[3\]. For the state-of-the-art subspace clustering algorithms, we use the codes provided by their authors. For SSC, we use the noisy variation of the optimization problem and choose \(\lambda_z = 20/\mu_z\) in all experiments. Following \[35\], we choose \(\lambda = 1/\sqrt{s}\) in RSSC. For TSC, we set \(q = \lceil N/(L \times 20) \rceil\). For the case of missing training data, we compare the results of rMiCUSaL with \(k\)-GROUSE \[24\] and GROUSE \[57\].

In order to study the robustness of our methods, we assume every training and test sample \(y\) is noisy in the sense that \(y = x + \xi\), where \(x\) belongs to one of the \(S_\ell\)'s (also \(\|x\|_2^2 = 1\)) and \(\xi\) is an independent Gaussian noise term with \(\mathcal{N}(0,(\sigma^2/m)I_m)\) distribution. We use \(\sigma^2_t\) and \(\sigma^2_{te}\) to denote the expected noise power of \(\xi\) \((\mathbb{E}[\|\xi\|_2^2] = \sigma^2)\) in the training and test data, respectively. The “clean” training and test signals are denoted by \(X\) and \(X^{te}\), respectively, while the set of noisy test samples is denoted by \(Y^{te}\). The Monte-Carlo simulations for noisy data are repeated 50 times for a fixed \(X\) and \(X^{te}\). In the missing data experiments, for every fixed noise power \(\sigma^2_t\), 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. In the experiments with synthetic data, we set the subspace dimension for PCA to be the (unrealizable) one which yields the best denoising result of test samples. In this setting, for a fixed \(\sigma_{te}\), the subspace dimensions for PCA will be different for different noise levels \(\sigma_{te}\)'s. Finally, we should note that for every fixed \(\sigma_t\) and \(\sigma_{te}\), we use the same dimension for GROUSE as in PCA in the corresponding missing data experiments.

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 \(m = 200\)-dimensional ambient space. The five subspaces \(S_\ell\)'s of \(\mathbb{R}^{200}\) are defined by their orthonormal bases \(\{T_\ell \in \mathbb{R}^{m \times s}\}_{\ell = 1}^5\) as follows. We start with a random orthonormal basis \(T_1 \in \mathbb{R}^{m \times s}\) and for every \(\ell \geq 2\), we set \(T_\ell = \text{orth}(T_{\ell-1} + t_s W_\ell)\) with \(W_\ell\) being a random Gaussian \(m \times s\) matrix, where \(\text{orth}(\cdot)\) 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 \(S_\ell\) as \(X_\ell = T_\ell C_\ell\), where \(C_\ell \in \mathbb{R}^{s \times n_\ell}\) is a matrix whose elements are drawn i.i.d. from \(\mathcal{N}(0,1)\) distribution. In this experiment, 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 \(X = [X_1, \ldots, X_5] = \{x_i\}_{i=1}^N\) and normalize all the samples to unit \(\ell_2\) norms. Test data \(X^{te} = \{x^{te}_i\}_{i=1}^N\) are produced using the same foregoing strategy. Then we add white Gaussian noise with different expected noise power to both \(X\) and \(X^{te}\). In the experiment, we set \(\sigma^2_t\) to be 0.05 and 0.1, while \(\sigma^2_{te}\) ranges from 0.1 to 0.5. We repeat the realizations of \(X\) and \(X^{te}\) 10 times. Therefore, the results reported in here correspond to an average of 500 Monte-Carlo trials.

Next, we make use of the collection of noisy samples, \(Y\), to learn a union of \(L\) subspaces of dimension \(s\) as discussed in \[4\], we omit the results for SSC with missing data in this paper because it fills in the missing entries with random values, resulting in its poor performance for such problems.
and stack the learned orthonormal bases \( \{D_\ell\}_{\ell=1}^L \) into \( D \). In this set of experiments, we perform MiCUSaL and rMiCUSaL for complete and missing data experiments, respectively. In order to learn these subspaces, we choose \( \lambda = 2 \) for all experiments. We use the following metric for MC-UoS learning performance analysis. Since we have knowledge of the ground truth \( S_\ell \)’s, represented by their ground truth orthonormal bases \( T_\ell \)’s, we first find the pairs of estimated/true subspaces that are the best match, i.e., \( D_\ell \) is matched to \( T_{\hat{\ell}} \) with \( \hat{\ell} = \arg \max_p \|D_\ell^T P_p\|_F \). We also

| Training noise level | Algorithms |
|----------------------|------------|
| \( \sigma^2_{\text{tr}} = 0.05 \) | MiCUSaL | SAC+BK-SVD | K-sub | SSC | RSSC | TSC |
| \( \sigma^2_{\text{tr}} = 0.1 \) | MiCUSaL | SAC+BK-SVD | K-sub | SSC | RSSC | TSC |

Fig. 1. Comparison of MC-UoS learning performance on synthetic data. (a) and (c) show relative errors of test signals for complete data experiment. (b) and (d) show relative errors of test signals for missing data experiment. The numbers in the legend of (b) and (d) indicate the percentages of missing entries in the training data.

TABLE I

d_{\text{avg}} OF DIFFERENT ALGORITHMS IN THE CASE OF COMPLETE SYNTHETIC DATA

| Training noise level | Algorithms |
|----------------------|------------|
| \( \sigma^2_{\text{tr}} = 0.05 \) | MiCUSaL | SAC+BK-SVD | K-sub | SSC | RSSC | TSC |
| \( \sigma^2_{\text{tr}} = 0.1 \) | MiCUSaL | SAC+BK-SVD | K-sub | SSC | RSSC | TSC |
ensure that no two $D_\ell$’s are matched to the same $T_\ell$. Then we define $d_{avg}$ to be the average normalized subspace distances between these pairs, i.e., $d_{avg} = \frac{1}{L} \sum_{\ell=1}^L \sqrt{\sum_{i=1}^N (D_\ell^T T_\ell^T D_\ell)^2}$. 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 would be 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 $y^{te}$ such that $y^{te} \approx D_\tau \alpha^{te} + \bar{y}$ where $\tau = \text{arg max}_\ell \|D_\ell^T (y^{te} - \bar{y})\|_2^2$ (recall that $\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$) and $\alpha^{te} = D_\tau^T y^{te}$. The relative reconstruction error with respect to $x^{te}$ is then defined as $\|x^{te} - (D_\tau \alpha^{te} + \bar{y})\|_2^2$. On the other hand, if the training data have missing entries then for a test signal $y^{te}$, the reconstruction error with respect to its noiseless part, $x^{te}$, is simply calculated by $\|x^{te} - D_\tau y^{te}\|_2^2$, where $\tau = \text{arg max}_\ell \|D_\ell^T y^{te}\|_2^2$.

Table I summarizes the $d_{avg}$’s for different union-of-subspaces learning algorithms for complete 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. I(a) and Fig. I(c) for a validation of this claim. For MC-UoS learning with missing data, rMiCUSaL also learns a better MC-UoS compared with k-GROUSE. First, the resulting $d_{avg}$’s for different number of missing entries of rMiCUSaL are around 0.16 when $\sigma_{tr}^2 = 0.05$ and 0.2 when $\sigma_{tr}^2 = 0.1$, respectively; while the corresponding $d_{avg}$’s for k-GROUSE are around 0.4 and 0.5, respectively. We can also infer from Fig. I(b) and Fig. I(d) that (i) rMiCUSaL outperforms k-GROUSE and GROUSE in terms of smaller reconstruction errors of test data; (ii) for a fixed $\sigma_{tr}$ and $\sigma_{tc}$, 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 very close to each other (e.g., $t_s = 0.2$). In such a case, all the UoS learning algorithms, including MiCUSaL, learn the subspaces successfully. We omit these plots because of space constraints.

In the second set of synthetic experiments, we examine the ability of aMiCUSaL to estimate $L$ given $Y$. Since

\footnote{Note that as the number of missing entries of training data increases, $d_{avg}$ will also increase. In here, we only give the range of $d_{avg}$’s for the amount of missing data ranging from 10\% to 50\% in the interest of space.}
the goal is to test whether aMiCUSaL can estimate the number of subspace correctly, we assume $s$ is known in this experiment. Hence we exclude dimensionality estimation step in aMiCUSaL and we term the resulting algorithm aMiCUSaL-Oracle in the following. The setup is similar to the previous experiments. The key parameters are $L = 2$, $s = 15$, $m = 150$, $n_1 = n_2 = 150$ (i.e., $N = 300$) and $t_s = 0.05$. To generate two subspaces, we start with a random orthonormal basis $T \in \mathbb{R}^{m \times s}$ and for every $\ell \in \{1, 2\}$, $T_\ell = \text{orth}(T + tW_\ell)$ with $W_\ell$ being a random Gaussian $m \times s$ matrix (note that $T_\ell$ is not generated from $T_{\ell - 1}$ at this time). We set $\sigma^2_{tr}$ to range from 0.02 to 0.1, while $\sigma^2_{te}$ is fixed to be 0.3. We also repeat the realizations of $X$ and $X^{te}$ 10 times. We set the parameters in aMiCUSaL-Oracle as $L_{\text{max}} = 5$, $\epsilon_{\text{min}} = 0.1$ and $\lambda = 1$.

First, we report the mean of the estimated $L$ of all 500 trials. As shown in Fig. 2(a) all the subspace clustering algorithms fail to estimate $L$ correctly and they all reduce to PCA. On the contrary, aMiCUSaL-Oracle will always give us the correct number of subspaces $L$ for low training noise levels $\sigma_{tr}$’s. Next, we compare aMiCUSaL-Oracle with PCA by evaluating their respective representation capability of test data. It can be seen from Fig. 2(b) that our method outperforms PCA at all noise levels $\sigma_{tr}$’s. While the gap between them decreases when $\sigma^2_{tr} = 0.08$ and 0.1. This is because aMiCUSaL-Oracle sometimes returns $L$ to be 3 at those higher $\sigma_{te}$’s.

2) Experiments on Real-World Image Data: To further show the effectiveness of the proposed approaches, we test our proposed methods on real-world image 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 $X$. Test signals in $X^{te} \in \mathbb{R}^{600 \times 722}$ are extracted in the same way from the right subimage. The Gaussian noise is then added to $X$ and $X^{te}$ separately, forming $Y$ and $Y^{te}$, respectively. In this experiment, $\sigma^2_{tr}$ is set to be 0.02 and 0.05, while $\sigma^2_{te}$ again ranges from 0.1 to 0.5. Note that each patch is treated as a single signal here, and our goal is learn an MC-UoS from $Y$ such that every test patch
Fig. 4. Comparison of MC-UoS learning performance on San Francisco City Hall data. (a) and (c) show relative errors of test signals for complete data experiment. (b) and (d) show relative errors of test signals for missing data experiment. The numbers in the legend of (b) and (d) indicate the percentages of missing entries in the training data.

can be denoised using the learned subspaces.

We perform aMiCUSaL on the training data $Y$ with parameters $L_{\text{max}} = 8$, $s_{\text{max}} = 20$, $\lambda = 4$, $k_1 = 6$, $k_2 = 10$ and $\epsilon_{\text{min}} = 0.08$. The results reported here correspond to an average of 50 trials. The output $L$ from aMiCUSaL is 4 or 5 and $s$ is always between 11 and 13. We also perform MiCUSaL with the same $L$ and $s$. For fair comparison, we use the method in this paper to also 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 subspace clustering 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 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 $L$ for TSC is sensitive to the noise and data. Specifically, the estimated $L$ is always 6 for $\sigma^2_{tr} = 0.02$ and $L$ is always even greater than 6 when $\sigma^2_{tr} = 0.05$, which results in poorer performance compared to the case when $L = 4$ or 5 for both training noise levels. For the missing data experiment, 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(c) 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. It can be inferred from Fig. 4(b) and Fig. 4(d) that rMiCUSaL also yields better representation performance for the missing data case.

Next, we repeated this experiment for the complete data experiment using Paris City Hall image in Fig. 3(b), forming $X, X_{te} \in \mathbb{R}^{600 \times 950}$. We perform aMiCUSaL using the same parameters as in the previous experiment. The estimated $L$ in this case ranges from 4 to 6 and $s$ is either 10 or 11. The estimated dimension of the subspace in PCA is always 10. In this experiment, we again use the same $L$ and $s$ as aMiCUSaL for all state-of-the-art subspace clustering algorithms. This is because the returned $L$ by SSC is again 1 in this case. The $L$ estimated by RSSC is usually from 6 to 9, but the reconstruction errors of test data is greater than the ones reported here. If we use $L$ generated by TSC itself, we will have very similar results since the estimated $L$ is 4 at this time. 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 the other algorithms. Moreover, since the estimated dimensionality of the subspaces in our method and in PCA are always the same, this suggests that the learned subspaces have better data approximation capability than a single subspace.

**B. Experiments for MC-KUoS Learning**

In this section, we evaluate the performance of our MC-KUoS learning approaches and report the denoising results on the USPS dataset, 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, we only use noiseless training data in this set of experiments. We assume every noisy test sample $y_{te} = x_{te} + \xi$, where $\phi(x_{te})$ belongs to one of the $S_t$’s in $\mathcal{F}$ (again $\|x_{te}\|_2^2 = 1$) and $\xi$ has $\mathcal{N}(0, (\sigma_{te}^2/m)I_m)$ distribution. We compare MC-KUSaL with three other methods: (i) kernel $k$-means clustering (kernel $k$-means) [17], (ii) kernel PCA [28] with the same number of eigenvectors as in MC-KUSaL (KPCA-Fix), and (iii) kernel PCA with the number of eigenvectors chosen by $s = \arg\min_s \|P_S\phi(y_{te}) - \phi(x_{te})\|_2^2$ (KPCA-Oracle), where $x_{te}$ and $y_{te}$ are clean and noisy test samples respectively. In this manner, the number of
Fig. 6. Comparison of MC-KUoS learning performance on USPS dataset using (a,b) Gaussian kernel $\kappa(y, y') = \exp(-\frac{\|y - y'\|_2^2}{8})$ and (c,d) polynomial kernel $\kappa(y, y') = (\langle y, y' \rangle + 1.5)^3$. Note that the KPCA-Oracle algorithm is the ideal case of kernel PCA. The numbers in the legend of (b) and (d) indicate the percentages of missing entries in the training data.

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 $x_{te}^i \in X^{te}$ is calculated by $\frac{\|x_{te}^i - \tilde{x}_{te}^i\|_2^2}{\|x_{te}^i\|_2^2}$, where $\tilde{x}_{te}^i$ denotes the pre-image with respect to the noisy test sample $y_{te}^i$.

We first experiment with Gaussian kernel with parameter $c = 8$. We choose the digits “1” and “6” and for each digit we select the last 200 samples in the dataset (400 images in total) in our experiments. All these 400 samples are then normalized to unit $\ell_2$ norms. Within these samples, we randomly choose 150 samples (without replacement) from each class for training and the remaining 50 samples of each class for testing, forming $X \in \mathbb{R}^{256 \times 300}$ and $X^{te} \in \mathbb{R}^{256 \times 100}$. This random selection of test and training samples is repeated 10 times for cross-validation purposes. We again repeat 50 Monte-Carlo trials for noisy test data and report the mean over these 500 trials.
In these experiments, we perform MC-KUSaL with parameters $L = 2$, $s = 45$ and $\lambda = 4$ to learn an MC-UoS in the feature space $F$. Fig. 6(a) shows the mean of relative reconstruction errors of test data for different methods in the presence of complete training data. We observe that for almost all noise levels, our method produces better results than other methods. The only exception is when $\sigma^2_{te} = 0.2$, in which case MC-KCUSaL 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.

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 $\lambda = 10$ for rMC-KUSaL. It can be inferred from Fig. 6(b) that (i) the performance of rKPCA and rMC-KUSaL is comparable for all noise levels; (ii) when the number of missing elements is fixed, rMC-KUSaL outperforms the rKPCA when the SNR of the test data is small and rKPCA outperforms the rMC-KUSaL when the noise level of the test data is relatively small.

Finally, we perform experiments with polynomial kernel $\kappa(y,y') = \langle y, y' \rangle + c$ with $c = 1.5$ and $d = 3$. We choose the first 200 samples of digits “0” and “1” in the USPS dataset and use these 400 samples in these experiments. The matrices $X$ and $X^{te}$ are generated in the same way as in the Gaussian kernel experiments. The parameters are $L = 2$, $s = 50$ and $\lambda = 1$ for both complete and missing data experiments. As described in Fig. 6(c), MC-KUSaL produces better results than other methods for all noise levels. In the missing data experiments, we set the number of missing entries in the training data to be only 2% and 5% of the signal dimension. From Fig. 6(d), we observe that as the number of missing entries is fairly small, we can have good reconstruction error of test data. But when we have 5% missing entries, the performance of both methods degrades a lot.

VI. CONCLUSION

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

APPENDIX

PROOF OF LEMMA

Proof: First, we have $\langle y_i, y_j \rangle = \sum_{u=1}^{m} z_{ij(u)}^*$ and $\langle [y_i]_{\Omega_{ij}}, [y_j]_{\Omega_{ij}} \rangle = \sum_{v=1}^{n} z_{ij(\Omega_{ij(v)})}^*$ with $n = |\Omega_{ij}|$. Here, $z_{ij(u)}^*$ denotes the $u$-th entry of a vector $z_{ij}^*$ and $\Omega_{ij(v)}$ denotes the $v$-th element of $\Omega_{ij}$. Let $h(Z_1, \ldots, Z_n) = \sum_{u=1}^{n} Z_u$ and $Z_v = z_{ij(\Omega_{ij(v)})}^*$. Hence $h$ is the sum of $n$ random variables. We prove the bound under the assumption that these $n$ variables are drawn uniformly from a set $\{z_{ij(1)}^*, \ldots, z_{ij(m)}^*\}$ with replacement. This means they are
independent and we have $E[\sum_{v=1}^{n} Z_v] = E[\sum_{v=1}^{n} z_{ij}(\Omega_{ij}(v))] = \frac{m}{n} \sum_{u=1}^{m} z_{ij}^*(u)$. If one variable in the sum is replaced by any other of its possible values then the sum changes at most $2\|z_{ij}^*\|_{\infty}$, i.e., $|\sum_{v=1}^{n} Z_v - \sum_{v \neq v'} Z_v - \tilde{Z}_{v'}| = |Z_{v'} - \tilde{Z}_{v'}| \leq 2\|z_{ij}^*\|_{\infty}$ for any $v' \in \{1, \ldots, n\}$. Therefore, McDiarmid’s Inequality \cite{4} shows that for $\beta > 0$,
\[
\Pr\left[\left|\sum_{v=1}^{n} Z_v - \frac{n}{m} \sum_{u=1}^{m} z_{ij}^*(u)\right| \geq \frac{n}{m} \beta\right] \leq 2 \exp\left(\frac{-n\beta^2}{2m^2\|z_{ij}^*\|_{\infty}^2}\right),
\]

or equivalently,
\[
\Pr\left[\sum_{u=1}^{m} z_{ij}^*(u) - \beta \leq \frac{m}{n} \sum_{v=1}^{n} Z_v \leq \sum_{u=1}^{m} z_{ij}^*(u) + \beta\right] \geq 1 - 2 \exp\left(\frac{-n\beta^2}{2m^2\|z_{ij}^*\|_{\infty}^2}\right).
\]

Taking the definition of $\beta = \sqrt{\frac{2m^2\|z_{ij}^*\|_{\infty}^2}{n\log(\frac{1}{\delta})}}$ yields the result. \hfill \□

\section*{References}

\begin{thebibliography}{10}

\bibitem{1} T. Wu and W. U. Bajwa, “Revisiting robustness of the union-of-subspaces model for data-adaptive learning of nonlinear signal models,” in \textit{Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP)}, 2014, pp. 3390–3394.
\bibitem{2} ——, “Subspace detection in a kernel space: The missing data case,” in \textit{Proc. IEEE Statistical Signal Processing Workshop (SSP)}, 2014, pp. 93–96.
\bibitem{3} H. Hotelling, “Analysis of a complex of statistical variables into principal components,” \textit{J. Educ. Psych.}, vol. 24, pp. 417–441, 1933.
\bibitem{4} S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” \textit{Science}, vol. 290, pp. 2323–2326, 2000.
\bibitem{5} T. Cox and M. Cox, \textit{Multidimensional scaling}. Chapman & Hall, 2000.
\bibitem{6} M. Elad, R. Goldenberg, and R. Kimmel, “Low bit-rate compression of facial images,” \textit{IEEE Trans. Image Process.}, vol. 16, no. 9, pp. 2379–2383, 2007.
\bibitem{7} R. G. Baraniuk, V. Cevher, and M. B. Wakin, “Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective,” \textit{Proc. IEEE}, vol. 98, no. 6, pp. 959–971, 2010.
\bibitem{8} M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” \textit{IEEE Trans. Image Process.}, vol. 15, no. 12, pp. 3736–3745, 2006.
\bibitem{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 \textit{Advances in Neural Information Processing Systems (NIPS)}, 1999, pp. 536–542.
\bibitem{10} M. A. Turk and A. P. Pentland, “Face recognition using eigenfaces,” in \textit{Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)}, 1991, pp. 586–591.
\bibitem{11} D. L. Swets and J. Weng, “Using discriminant eigenfeatures for image retrieval,” \textit{IEEE Trans. Pattern Anal. Mach. Intell.}, vol. 18, no. 8, pp. 831–836, 1996.
\bibitem{12} J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” \textit{IEEE Trans. Pattern Anal. Mach. Intell.}, vol. 31, no. 2, pp. 210–227, 2009.
\bibitem{13} J. Mairal, F. Bach, and J. Ponce, “Task-driven dictionary learning,” \textit{IEEE Trans. Pattern Anal. Mach. Intell.}, vol. 34, no. 4, pp. 791–804, 2012.
\bibitem{14} S. Rao, R. Tron, R. Vidal, and Y. Ma, “Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories,” \textit{IEEE Trans. Pattern Anal. Mach. Intell.}, vol. 32, no. 10, pp. 1832–1845, 2010.
\bibitem{15} E. Elhamifar and R. Vidal, “Sparse subspace clustering: Algorithm, theory, and applications,” \textit{IEEE Trans. Pattern Anal. Mach. Intell.}, vol. 35, no. 11, pp. 2765–2781, 2013.
\bibitem{16} H. H. Harman, \textit{Modern factor analysis}. University of Chicago Press, 1976.
\bibitem{17} B. Schölkopf, A. J. Smola, and K. R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” \textit{Neural Comput.}, vol. 10, no. 5, pp. 1299–1319, 1998.
\end{thebibliography}
[18] 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.

[19] 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.

[20] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, “An overview of JPEG-2000,” in *Proc. IEEE Data Compression Conf.*, 2000, pp. 523–541.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

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

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

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

[31] 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.

[32] 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.

[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. Statist.*, vol. 40, no. 4, pp. 2195–2238, 2012.

[35] M. Soltanolkotabi and E. Elhamifar, and E. J. Candès, “Robust subspace clustering,” *arXiv:1301.2603*, 2013.

[36] 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.

[37] R. Heckel and H. Bölcskei, “Robust subspace clustering via thresholding,” *arXiv:1307.4891*, 2013.

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] X. Sun, L. Wang, and J. Feng, “Further results on the subspace distance,” *Pattern Recognition*, vol. 40, no. 1, pp. 328–329, 2007.
[44] 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.
[45] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in Proc. Intl. Joint Conf. Artificial Intelligence (IJCAI), 1995, pp. 1137–1143.
[46] 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.
[47] J. C. Bezdek and R. J. Hathaway, “Convergence of alternating optimization,” Neural, Parallel Sci. Comput., vol. 11, no. 4, pp. 351–368, 2003.
[48] D. P. Bertsekas, Nonlinear programming. Athena Scientific, 1999.
[49] E. Kokio pou lou, 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.
[50] 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.
[51] 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.
[52] S. Kritchman and B. Nadler, “Determining the number of components in a factor model from limited noisy data,” Chem. Int. Lab. Syst., vol. 94, no. 1, pp. 19–32, 2008.
[53] E. Levina and P. J. Bickel, “Maximum likelihood estimation of intrinsic dimension,” in Advances in Neural Information Processing Systems (NIPS), 2004, pp. 777–784.
[54] L. Balzano, B. Recht, and R. Nowak, “High-dimensional matched subspace detection when data are missing,” in Proc. IEEE Intl. Symp. Inf. Theory (ISIT), 2010, pp. 1638–1642.
[55] 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, 1999.
[56] A. Nedi ´c and D. P. Bertsekas, “Incremental subgradient methods for nondifferentiable optimization,” SIAM J. Optim., vol. 12, no. 1, pp. 109–138, 2001.
[57] 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 (Allerton), 2010, pp. 704–711.
[58] B. Eriksson, L. Balzano, and R. Nowak, “High-rank matrix completion,” in Proc. Conf. Artificial Intelligence and Statistics (AISTATS), 2012, pp. 373–381.
[59] C. J. C. Burges, “Simplified support vector decision rules,” in Proc. Intl. Conf. Machine Learning (ICML), 1996, pp. 71–77.
[60] Y. Rathi, S. Dambreville, and A. Tannenbaum, “Statistical shape analysis using kernel PCA,” in Proc. SPIE, vol. 6064, 2006, pp. 425–432.
[61] C. McDiarmid, “On the method of bounded differences,” Surveys in Combinatorics, vol. 141, pp. 148–188, 1989.