LEARNING THE NONLINEAR GEOMETRIC STRUCTURE OF HIGH-DIMENSIONAL DATA: MODELS, ALGORITHMS, AND APPLICATIONS

BY TONG WU

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Prof. Waheed U. Bajwa

and approved by

New Brunswick, New Jersey

May, 2017

ABSTRACT OF THE DISSERTATION

Learning the nonlinear geometric structure of high-dimensional data: Models, Algorithms, and Applications

by TONG WU

Dissertation Director: Prof. Waheed U. Bajwa

Modern information processing relies on the axiom that high-dimensional data lie near low-dimensional geometric structures. The work presented in this thesis aims to develop new models and algorithms for learning the geometric structures underlying data and to exploit the application of geometry learning in image and video analytics.

The first part of the thesis revisits the problem of data-driven learning of these geometric structures and puts forth two new nonlinear geometric models for data describing "related" objects/phenomena. The first one of these models straddles the two extremes of the subspace model and the union-of-subspaces model, and is termed the *metric-constrained union-of-subspaces* (MC-UoS) model. The second one of these models—suited for data drawn from a mixture of nonlinear manifolds—generalizes the kernel subspace model, and is termed the *metric-constrained kernel union-of-subspaces* (MC-KUoS) model. The main contributions in this regard are threefold. First, we motivate and formalize the problems of MC-UoS and MC-KUoS learning. Second, we present algorithms that efficiently learn an MC-UoS or an MC-KUoS underlying data of interest. Third, we extend these algorithms to the case when parts of the data are missing.

The second part of the thesis considers the problem of learning meaningful human action attributes from video data. Representation of human actions as a sequence of human body movements or action attributes enables the development of models for human activity recognition and summarization. We first propose a hierarchical union-of-subspaces model and an approach called hierarchical sparse subspace clustering (HSSC) is developed to learn this model from the data in an unsupervised manner by capturing the variations or movements of each action in different subspaces. We then present an extension of the low-rank representation (LRR) model, termed the *clustering-aware structure-constrained low-rank representation* (CS-LRR) model, for unsupervised learning of human action attributes from video data. The CS-LRR model is based on the union-of-subspaces framework, and integrates spectral clustering into the LRR optimization problem for better subspace clustering results. We also introduce a hierarchical subspace clustering approach, termed hierarchical CS-LRR, to learn the attributes without the need for a priori specification of their number. By visualizing and labeling these action attributes, the hierarchical model can be used to semantically summarize long video sequences of human actions at multiple resolutions. A human action or activity can also be uniquely represented as a sequence of transitions from one action attribute to another, which can then be used for human action recognition.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my PhD advisor, Prof. Waheed U. Bajwa, for his guidance and support during the past five years. Ever since the first few days when he guided me through my first research project, Prof. Bajwa has been an outstanding advisor and friend. His enthusiasm for research, constant encouragement, and strong willingness to support his students' growth and independence always inspire me in both research work and personal life. It is my great fortune to work with him for five years and this experience will be the most cherished treasure of my life.

Next, I would like to express my sincerest gratitude to Dr. Prudhvi Gurram at U.S. Army Research Lab, for his advice and generous help on a number of academic and nonacademic matters during our collaborative research. I would also like to thank the other three members of my dissertation committee, Prof. Vishal Patel, Prof. Laleh Najafizadeh and Dr. Raghuveer M. Rao, for their precious time in reviewing my dissertation and providing valuable suggestions. I am also grateful to Prof. Kristin Dana, who served on my dissertation proposal committee.

In addition, I would like to thank all the members of the Information, Networks, and Signal Processing Research (INSPIRE) Lab, including Haroon Raja, Talal Ahmed, Zahra Shakeri, Muhammad Asad Lodhi and Zhixiong Yang, for their help and friendship.

Finally, I want to thank my parents for their unconditional support and continuous encouragement. Thank you for your sacrifices and constant support during my years of study in the United States.

Dedication

To my parents, for their endless love, support and encouragement.

Table of Contents

Chapter 1

Introduction

We have witnessed an explosion in data generation in the last decade or so. Modern signal processing, machine learning and statistics have been relying on a fundamental maxim of information processing to cope with this data explosion. This maxim states that while real-world data might lie in a high-dimensional Hilbert space, relevant information within them almost always lies near low-dimensional geometric structures embedded in the Hilbert space. Knowledge of these low-dimensional geometric structures not only improves the performance of many processing tasks, but 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 [1–5], denoising [6, 7], classification [8–11], and motion segmentation [12, 13]. These geometric models broadly fall into two categories, namely, linear models [1, 9, 14] and nonlinear models [3, 13, 15–17]. A further distinction can be made within each of these two categories depending upon whether the models are prespecified [18, 19] or learned from the data themselves [7, 13, 16, 20–22]. The focus in this thesis is on the latter case, since data-driven learning of geometric models is known to outperform prespecified geometric models [7, 23].

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) [1, 24], Karhunen–Loève transform [25], factor analysis [14], etc. But real-world data in many applications tend to be nonlinear. In order to better capture the geometry of data in such applications, a few nonlinear generalizations of data-driven linear models that remain computationally feasible have been investigated in the last two decades. One of the most popular generalizations is the nonlinear manifold model [3, 5, 26, 27]. The (nonlinear) manifold model can also be considered as the *kernel subspace* model, which dictates that a mapping of the data to a higher- (possibly infinite-) dimensional Hilbert space lies near a low-dimensional subspace [28]. Data-driven learning of geometric models in this case is commonly studied under the moniker of *kernel PCA* (KPCA) [26]. Another one of the most popular generalizations of linear models is the *union-of-subspaces* (UoS) (resp., *union-of-affine-subspaces* (UoAS)) model, which dictates that data lie near a mixture of low-dimensional subspaces (resp., affine subspaces) in the ambient Hilbert space. Data-driven learning of the UoS model is commonly carried out under the rubrics of generalized PCA [29], dictionary learning [16, 30], and subspace clustering [13, 31–34]. On the other hand, data-driven learning of the UoAS model is often studied under the umbrella of hybrid linear modeling [35], mixture of factor analyzers [36], etc.

1.1 Thesis Statement

In this thesis, we first consider the problem of learning the geometric structure underlying the signals describing similar phenomenon (e.g., similar frontal facial images). 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 [6, 11, 13, 37]. But there remains a lot of room for improvement in both these models. The canonical UoS model, for example, does not impose any constraint on the collection of subspaces underlying data of interest. On the other hand, one can intuit that subspaces describing "similar" data should have some "relation" on the Grassmann manifold. The lack of any a priori constraint during learning on the subspaces describing "similar" data has the potential to make different methods for UoS learning susceptible to errors due to low *signal-to-noise ratio* (SNR), outliers, missing data, etc. Another limitation of the UoS model is the individual linearity of its constituent subspaces, which limits its usefulness for data drawn from a nonlinear manifold [26]. On the other hand, while the kernel subspace model can handle manifold data, a single kernel subspace requires a large dimension to capture the richness of data drawn from a mixture of nonlinear manifolds.

Our first goal in this thesis is to improve the state-of-the-art data-driven learning of geometric data models for both complete and missing data describing similar phenomenon. We are in particular interested in learning models for data that are either mildly or highly nonlinear. Here, we are informally using the terms "mildly nonlinear" and "highly nonlinear." Heuristically, nonlinear data that cannot be represented through a mixture of linear components should be deemed "highly nonlinear." Our key objective in this regard is overcoming the aforementioned limitations of the UoS model and the kernel subspace model for mildly nonlinear data and highly nonlinear data, respectively. The solution to these two problems are addressed in detail in Chapter 2 and Chapter 3, respectively.

Next, we consider the problem of learning meaningful human action attributes from video data. A complex human activity or a high-level event can be considered to be a hierarchical model [38], consisting of a sequence of simpler human actions. Each action can further be divided into a sequence of movements of the human body, which we call action attributes [39]. In the past, a significant fraction of the video analytics literature has been devoted to the study of learning attributes in human actions [40,41]. In this thesis, we propose to represent human action attributes based on the UoS model [13,42], which is motivated by the fact that high-dimensional video data usually lie in a union of low-dimensional subspaces instead of being uniformly distributed in the high-dimensional ambient space [13]. The hypothesis of this model is that each action attribute is represented by a subspace. A human action or activity can then be represented as a sequence of transitions from one attribute to another and, hence, can be represented by a subspace transition vector. Even though multiple actions can share action attributes, each action or activity can be uniquely represented by its subspace transition vector when the attributes are learned using the UoS model. One of the applications of learning the subspaces based on the UoS model is *semantic description of long video sequences* with multiple human actions. Since the subspaces corresponding to each action attribute can be visualized using the first few dimensions of the corresponding orthonormal bases and assigned a semantic label, any long video sequence can be semantically explained using the semantic labels of the subspaces to which the frames are assigned and the transitions between the subspaces. Another major application of this representation is *human action recognition*. If training labels are available for human actions, classifiers can be trained for each of the actions based on the transition sequence and can be used to recognize human actions in videos. We describe the details of learning the attributes from video data in Chapter 4 and Chapter 5.

1.2 Major Contributions

In this thesis, we present new frameworks for learning the geometric structure underlying imaging/video data. We propose algorithms based on union-of-subspaces model and optimization techniques to effectively address different problems such as clustering and human action attribute learning. Below, we highlight some of the primary aspects of the thesis contributions.

1.2.1 Metric-Constrained Union-of-Subspaces

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

One of our other main contributions is extension of our MC-UoS model for highly nonlinear data. This model, which can also be considered a generalization of the kernel subspace model, is termed *metric-constrained kernel union-of-subspaces* (MC-KUoS) model. The MC-KUoS model asserts that mapping of data describing similar phenomenon to some higher-dimensional Hilbert space (also known as the *feature space*) lies near a mixture of subspaces in the feature space with the additional constraint that the individual subspaces are also close to each other in the feature space. In this regard, we formulate the MC-KUoS learning problem using the *kernel trick* [15], 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.

1.2.2 Human Action Attribute Learning

In the second part of the thesis, we address the problem of learning meaningful human action attributes from video data. A cornerstone of this effort in this regard is union-of-subspaces model for learning of low- and medium-level features from video sequences. In Chapter 4, we propose a hierarchical union-of-subspaces model to learn human action attributes and use Sparse Subspace Clustering (SSC) [13], a state-of-the-art subspace clustering method on top of it. We use silhouette structure of the human as the feature in this framework. Experimental results demonstrate the hierarchical UoS model results in better performance than the canonical UoS models in terms of human action recognition.

In Chapter 5, we introduce an extension of the low-rank representation (LRR) model, termed *clustering-aware structure-constrained LRR* (CS-LRR) model, to obtain optimal clustering of human action attributes from a large collection of video sequences. We formulate the CS-LRR learning problem by introducing spectral clustering into the optimization program. We also propose a hierarchical extension of our CS-LRR model for unsupervised learning of human action attributes from the data at different resolutions without assuming any knowledge of the number of attributes present in the data. Once the graph is learned from CS-LRR model, we segment it by applying hierarchical spectral clustering to obtain action attributes at different resolutions. The proposed approach is called *hierarchical clustering-aware structure-constrained LRR* (HCS-LRR). Our results confirm the superiority of HCS-LRR in comparison to a number of state-of-the-art subspace clustering approaches.

1.3 Notational Convention

The following notation will be used throughout the rest of this thesis. We use non-bold letters to represent scalars, bold lowercase letters to denote vectors/sets, and bold uppercase letters to denote matrices. The i-th element of a vector $\mathbf{a}$ is denoted by $a_{(i)}$ and the (i,j)-th element of a matrix $\mathbf{A}$ is denoted by $a_{i,j}$. The i-th row and j-th column of a matrix $\mathbf{A}$ are denoted by $\mathbf{a}^i$ and $\mathbf{a}_j$, respectively. Given a set $\mathbf{\Omega}$, $[\mathbf{A}]_{\mathbf{\Omega},:}$ (resp., $[\mathbf{v}]_{\mathbf{\Omega}}$) denotes the submatrix of $\mathbf{A}$ (resp., subvector of $\mathbf{v}$) corresponding to the rows of $\mathbf{A}$ (resp., entries of $\mathbf{v}$) indexed by $\mathbf{\Omega}$. Given two sets $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$, $[\mathbf{A}]_{\mathbf{\Omega}_1,\mathbf{\Omega}_2}$ denotes the submatrix of $\mathbf{A}$ corresponding to the rows and columns indexed by $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$, respectively. The zero matrix and the identity matrix are denoted by $\mathbf{0}$ and $\mathbf{I}$ of appropriate dimensions, respectively.

The most commonly used vector norm in this thesis is the ℓ_2 norm, which is represented by $\|\cdot\|_2$. We use a variety of norms on matrices. The ℓ_1 and $\ell_{2,1}$ norms are denoted by $\|\mathbf{A}\|_1 = \sum_{i,j} |a_{i,j}|$ and $\|\mathbf{A}\|_{2,1} = \sum_j \|\mathbf{a}_j\|_2$, respectively. The ℓ_∞ norm is defined as $\|\mathbf{A}\|_\infty = \max_{i,j} |a_{i,j}|$. The spectral norm of a matrix $\mathbf{A}$, i.e., the largest singular value of $\mathbf{A}$, is denoted by $\|\mathbf{A}\|$. The Frobenius norm and the nuclear norm (the sum of singular values) of a matrix $\mathbf{A}$ are denoted by $\|\mathbf{A}\|_F$ and $\|\mathbf{A}\|_*$, respectively. Finally, the Euclidean inner product between two matrices is $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{tr}(\mathbf{A}^T\mathbf{B})$, where $(\cdot)^T$ and $\mathrm{tr}(\cdot)$ denote transpose and trace operations, respectively.

1.4 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2, we formally define the metric-constrained union-of-subspaces (MC-UoS) model and present three algorithms for learning the geometry of mildly nonlinear data. In Chapter 3, we extend the MC-UoS model to the feature space and give the details of two algorithms for learning of an MC-UoS in the feature space, corresponding to the cases of complete and missing data. In Chapter 4, we present our hierarchical union-of-subspaces model and an algorithm for learning the human action attributes for video summarization and human action recognition. In Chapter 5, we describe the CS-LRR model and present an algorithm based on CS-LRR model. We also extend the CS-LRR model into a hierarchical structure. Finally, we conclude this thesis and discuss future work in Chapter 6.

Chapter 2

Metric-Constrained Union-of-Subspaces

2.1 Problem Formulation

In this chapter, we study the problem of learning the geometry of mildly nonlinear data based on the metric-constrained union-of-subspaces (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 [5, 43]: $\mathcal{M}_L = \bigcup_{\ell=1}^{L} \mathcal{S}_\ell$, where $\mathcal{S}_\ell$ is a subspace of $\mathbb{R}^m$. In here, we make the simplified assumption that all subspaces in $\mathcal{M}_L$ have the same dimension, i.e., $\forall \ell$, $\dim(\mathcal{S}_\ell) = s \ll m$. In this case, each subspace $\mathcal{S}_\ell$ corresponds to a point on the Grassmann manifold $\mathcal{G}_{m,s}$, which denotes the set of all s-dimensional subspaces of $\mathbb{R}^m$. While the canonical UoS model allows $\mathcal{S}_\ell$'s to be arbitrary points on $\mathcal{G}_{m,s}$, the basic premise of the MC-UoS model is that subspaces underlying *similar* signals likely form a "cluster" on the Grassmann manifold. In order to formally capture this intuition, we make use of a distance metric on $\mathcal{G}_{m,s}$ and define an MC-UoS according to that metric as follows.

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

The metric we use to measure distances between subspaces is based on the Hausdorff distance between a vector and a subspace, which was first defined in [44]. Specifically, if $\mathbf{D}_\ell \in \mathbb{R}^{m\times s}$ and $\mathbf{D}_p \in \mathbb{R}^{m\times s}$ denote orthonormal bases of subspaces $\mathcal{S}_\ell$ and $\mathcal{S}_p$, respectively, then

$$d_u(\mathcal{S}_\ell, \mathcal{S}_p) = \sqrt{s - \operatorname{tr}(\mathbf{D}_\ell^T \mathbf{D}_p \mathbf{D}_p^T \mathbf{D}_\ell)} = \|\mathbf{D}_\ell - P_{\mathcal{S}_p} \mathbf{D}_\ell\|_F, \tag{2.1}$$

where $P_{\mathcal{S}_p}$ denotes the projection operator onto the subspace $\mathcal{S}_p$: $P_{\mathcal{S}_p} = \mathbf{D}_p\mathbf{D}_p^T$. It is easy to convince oneself that $d_u(\cdot,\cdot)$ in (2.1) is invariant to the choice of orthonormal bases of the two subspaces, while it was formally shown to be a metric on $\mathcal{G}_{m,s}$ in [45]. Note that $d_u(\cdot,\cdot)$ in (2.1) is directly related to the concept of *principal angles* between two subspaces. Given two subspaces $\mathcal{S}_\ell, \mathcal{S}_p$ and their orthonormal bases $\mathbf{D}_\ell, \mathbf{D}_p$, the cosines of the principal angles $\cos(\theta_{\ell,p}^j)$, $j = 1, \dots, s$, between $\mathcal{S}_\ell$ and $\mathcal{S}_p$ are defined as the ordered singular values of $\mathbf{D}_\ell^T\mathbf{D}_p$ [34]. It therefore follows that $d_u(\mathcal{S}_\ell, \mathcal{S}_p) = \sqrt{s - \sum_{j=1}^s \cos^2(\theta_{\ell,p}^j)}$. We conclude our discussion of the MC-UoS model by noting that other definitions of metrics on the Grassmann manifold exist in the literature that are based on different manipulations of $\cos(\theta_{\ell,p}^j)$'s [46]. In this work, however, we focus only on (2.1) due to its ease of computation. Next, we assume access to a collection of N noisy training samples, $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N] \in \mathbb{R}^{m\times N}$, such that every sample $\mathbf{y}_i$ can be expressed as $\mathbf{y}_i = \mathbf{x}_i + \boldsymbol{\xi}_i$ with $\mathbf{x}_i$ belonging to one of the $\mathcal{S}_\ell$'s in $\mathcal{M}_L$ and $\boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{0}, (\sigma_{tr}^2/m)\mathbf{I}_m)$ denoting additive noise. We assume without loss of generality throughout this chapter that $\|\mathbf{x}_i\|_2^2 = 1$, which results in training SNR of $\|\mathbf{x}_i\|_2^2/\mathbb{E}[\|\boldsymbol{\xi}_i\|_2^2] = \sigma_{tr}^{-2}$. To begin, we assume both L and s are known a priori. Later, we relax this assumption and extend our work in Section 2.3 to the case when these two parameters are unknown. Our goal is to learn $\mathcal{M}_L$ using the training data $\mathbf{Y}$, which is equivalent to learning a collection of L subspaces that not only approximate the training data, but are also "close" to each other on the Grassmann manifold (cf. Definition 1). Here, we pose this goal of learning an MC-UoS $\mathcal{M}_L$ in terms of the following optimization program:

$$\{\mathcal{S}_\ell\}_{\ell=1}^L = \underset{\{\mathcal{S}_\ell\}\subset\mathcal{G}_{m,s}}{\arg\min} \sum_{\substack{\ell,p=1\\ \ell\neq p}}^{L} d_u^2(\mathcal{S}_\ell, \mathcal{S}_p) + \lambda \sum_{i=1}^{N} \|\mathbf{y}_i - P_{\mathcal{S}_{l_i}}\mathbf{y}_i\|_2^2, \tag{2.2}$$

where $l_i = \arg\min_\ell \|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ with $P_{\mathcal{S}_\ell}\mathbf{y}_i$ denoting the (orthogonal) projection of $\mathbf{y}_i$ onto the subspace $\mathcal{S}_\ell$. Notice that the first term in (2.2) 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 λ is beyond the scope of this work, cross validation can be used to find ranges of good values of tuning parameters in such problems [47] (also, see Section 2.5). It is worth pointing out here that (2.2) can be reformulated for the UoAS model through a simple extension of the metric defined in (2.1). In addition, note that (2.2) is mathematically similar to a related problem studied in the clustering literature [48]. In fact, it is straightforward to show that (2.2) reduces to the clustering problem in [48] for $\mathcal{M}_L$ being a union of zero-dimensional affine subspaces.

Remark 1. The MC-UoS model and the learning problem (2.2) can be further motivated as follows. Consider a set of facial images of individuals under varying illumination conditions in the Extended Yale B dataset [49], as in Figs. 2.1(a) and 2.1(b). It is generally agreed that all images of an individual in this case can be regarded as lying near a 9-dimensional subspace [50], which can be computed in a straightforward manner using singular value decomposition (SVD). The subspace distance defined in (2.1) can be used in this case to identify similar-looking individuals. Given noisy training images of such "similar" individuals, traditional methods for UoS learning such as *sparse subspace clustering* (SSC) [13] that rely only on the approximation error will be prone to errors. Fig. 2.1 provides a numerical validation of this claim, where it is shown that SSC has good performance on noisy images of different-looking individuals (cf. Fig. 2.1(b)), but its performance degrades in the case of similar-looking individuals (cf. Fig. 2.1(a)). The MC-UoS learning problem (2.2), on the other hand, should be able to handle both cases reliably because of the first term in (2.2) that penalizes subspaces that do not cluster on the Grassmann manifold. We refer the reader to Section 2.5.3 for detailed experiments that numerically validate this claim.

In this work, we study two variants of the MC-UoS learning problem described by (2.2). In the first variant, all m dimensions of each training sample in $\mathbf{Y}$ are observed and the geometry learning problem is exactly given by (2.2). In the second variant, it is assumed that some of the m dimensions of each training sample in $\mathbf{Y}$ are unobserved (i.e., missing), which then requires a recharacterization of (2.2) for the learning problem to be well posed. We defer that recharacterization to Section 2.4. In order to quantify the performance of our learning algorithms, we will resort to generation of noisy test data as follows. Given noiseless (synthetic or real) data sample $\mathbf{x}$ with $\|\mathbf{x}\|_2^2 = 1$, noisy test sample $\mathbf{z}$ is given by $\mathbf{z} = \mathbf{x} + \boldsymbol{\xi}$ with the additive noise $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, (\sigma_{te}^2/m)\mathbf{I}_m)$.

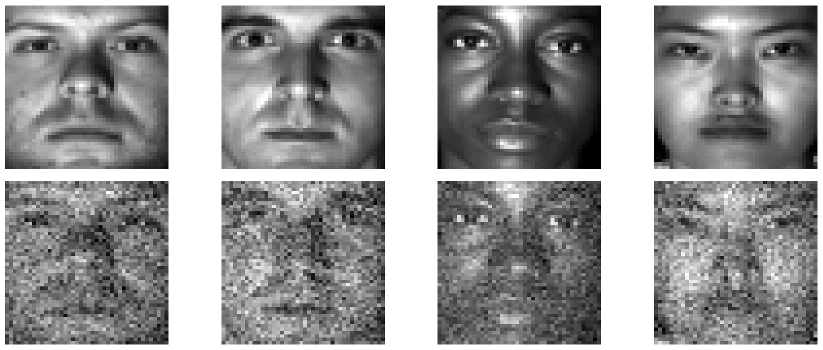


Figure 2.1: An example illustrating the limitations of existing methods for UoS learning from noisy training data. The top row in this figure shows examples of “clean” facial images of four individuals in the Extended Yale B dataset [49], while the bottom row shows noisy versions of these images, corresponding to $\sigma_{tr}^2 = 0.1$. The “ground truth” distance between the subspaces of the individuals in (a) is 1.7953, while it is 2.3664 between the subspaces of the individuals in (b). State-of-the-art UoS learning methods have trouble reliably learning the underlying subspaces whenever the subspaces are close to each other. Indeed, while the distance between the two subspaces learned by the SSC algorithm [13] from noisy images of the individuals in (b) is 2.4103, it is 2.4537 for the case of “similar-looking” individuals in (a).

We will then report the metric of *average approximation error of noisy test data* using the learned subspaces for synthetic and real data. Finally, in the case of synthetic data drawn from an MC-UoS, we will also measure the performance of our algorithms in terms of *average normalized subspace distances* between the learned and the true subspaces. We defer a formal description of both these metrics to Section 2.5.1, which describes in detail the setup of our experiments.

2.2 MC-UoS Learning Using Complete Data

In order to reduce the effects of noisy training data, we begin with a pre-processing step that centers the data matrix $\mathbf{Y}$.[1] This involves defining the mean of the samples in $\mathbf{Y}$ as $\bar{\mathbf{y}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i$ and then subtracting this mean from $\mathbf{Y}$ to obtain the centered data $\widetilde{\mathbf{Y}} = [\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_N]$, where $\widetilde{\mathbf{y}}_i = \mathbf{y}_i - \bar{\mathbf{y}}$, $i = 1, \ldots, N$. Next, we focus on simplification of the optimization problem (2.2). To this end, we first define an $L \times N$ indicator

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

matrix $\mathbf{W}$ that identifies memberships of the $\widetilde{\mathbf{y}}_i$'s in different subspaces, where $w_{\ell,i} = 1$, $\ell = 1, \dots, L$, $i = 1, \dots, N$, if and only if $\widetilde{\mathbf{y}}_i$ is "closest" to subspace $\mathcal{S}_\ell$; otherwise, $w_{\ell,i} = 0$. Mathematically,

$$\mathbf{W} = \big[w_{\ell,i} \in \{0,1\} : \forall i = 1, \dots, N, \sum_{\ell=1}^{L} w_{\ell,i} = 1\big]. \tag{2.3}$$

Further, notice that $\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ in (2.2) can be rewritten as

$$\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2 = \|\widetilde{\mathbf{y}}_i - P_{\mathcal{S}_\ell}\widetilde{\mathbf{y}}_i\|_2^2 = \|\widetilde{\mathbf{y}}_i\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\mathbf{y}}_i\|_2^2, \tag{2.4}$$

where $\mathbf{D}_\ell \in \mathbb{R}^{m\times s}$ denotes an (arbitrary) orthonormal basis of $\mathcal{S}_\ell$. Therefore, defining $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_L]$ to be a collection of orthonormal bases of $\mathcal{S}_\ell$'s, we can rewrite (2.2) as $(\mathbf{D}, \mathbf{W}) = \arg\min_{\mathbf{D},\mathbf{W}} F_1(\mathbf{D}, \mathbf{W})$ with the objective function $F_1(\mathbf{D}, \mathbf{W})$ given by[2]

$$F_1(\mathbf{D}, \mathbf{W}) = \sum_{\substack{\ell,p=1\\ \ell\neq p}}^{L} \|\mathbf{D}_\ell - P_{\mathcal{S}_p}\mathbf{D}_\ell\|_F^2 + \lambda\sum_{i=1}^{N}\sum_{\ell=1}^{L} w_{\ell,i}(\|\widetilde{\mathbf{y}}_i\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\mathbf{y}}_i\|_2^2). \tag{2.5}$$

Minimizing (2.5) simultaneously over $\mathbf{D}$ and $\mathbf{W}$ is challenging and is likely to be computationally infeasible. Instead, we adopt an alternate minimization approach [51, 52], which involves iteratively solving (2.5) by alternating between the following two steps: (i) minimizing $F_1(\mathbf{D}, \mathbf{W})$ over $\mathbf{W}$ for a fixed $\mathbf{D}$, which we term as the *subspace assignment* step; and (ii) minimizing $F_1(\mathbf{D}, \mathbf{W})$ over $\mathbf{D}$ for a fixed $\mathbf{W}$, which we term as the *subspace update* stage. To begin this alternate minimization, we start with an initial $\mathbf{D}$ in which each block $\mathbf{D}_\ell \in \mathbb{R}^{m\times s}$ is a random orthonormal basis. Next, we fix this $\mathbf{D}$ and carry out subspace assignment, which now amounts to solving for each $i = 1, \dots, N$,

$$l_i = \underset{\ell=1,\dots,L}{\arg\min} \|\widetilde{\mathbf{y}}_i - P_{\mathcal{S}_\ell}\widetilde{\mathbf{y}}_i\|_2^2 = \underset{\ell=1,\dots,L}{\arg\max} \|\mathbf{D}_\ell^T\widetilde{\mathbf{y}}_i\|_2^2, \tag{2.6}$$

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

[2]Note that the minimization here is being carried out under the assumption of $\mathbf{D}_\ell$'s being orthonormal and $\mathbf{W}$ being described by (2.3).

resorting to block-coordinate descent (BCD) [51] and updating only one $\mathbf{D}_\ell$ at a time while keeping the other $\mathbf{D}_p$'s ($p \neq \ell$) fixed in (2.5). In this regard, suppose we are in the process of updating $\mathbf{D}_\ell$ for a fixed ℓ during the subspace update step. Define $\mathbf{c}_\ell = \{i \in \{1, \dots, N\} : w_{\ell,i} = 1\}$ to be the set containing the indices of all $\widetilde{\mathbf{y}}_i$'s that are assigned to $\mathcal{S}_\ell$ (equivalently, $\mathbf{D}_\ell$) and let $\widetilde{\mathbf{Y}}_\ell = [\widetilde{\mathbf{y}}_i : i \in \mathbf{c}_\ell]$ be the corresponding $m \times |\mathbf{c}_\ell|$ matrix. Then it can be shown after some manipulations of (2.5) that updating $\mathbf{D}_\ell$ is equivalent to solving the following problem:

$$\begin{aligned}\mathbf{D}_\ell &= \operatorname*{arg\,min}_{\mathbf{Q} \in \mathcal{V}_{m,s}} \sum_{p \neq \ell} \|\mathbf{Q} - P_{\mathcal{S}_p}\mathbf{Q}\|_F^2 + \frac{\lambda}{2}(\|\widetilde{\mathbf{Y}}_\ell\|_F^2 - \|\mathbf{Q}^T\widetilde{\mathbf{Y}}_\ell\|_F^2) \\ &= \operatorname*{arg\,max}_{\mathbf{Q} \in \mathcal{V}_{m,s}} \operatorname{tr}\Big(\mathbf{Q}^T(\sum_{p \neq \ell} \mathbf{D}_p\mathbf{D}_p^T + \frac{\lambda}{2}\widetilde{\mathbf{Y}}_\ell\widetilde{\mathbf{Y}}_\ell^T)\mathbf{Q}\Big), \end{aligned} \tag{2.7}$$

where $\mathcal{V}_{m,s}$ denotes the Stiefel manifold, defined as the collection of all $m \times s$ orthonormal matrices. Note that (2.7) has an intuitive interpretation. When $\lambda = 0$, (2.7) reduces to the problem of finding a subspace that is closest to the remaining $L-1$ subspaces in our collection. When $\lambda = \infty$, (2.7) reduces to the PCA problem, in which case the learning problem (2.2) reduces to the subspace clustering problem studied in [53]. By selecting an appropriate $\lambda \in (0, \infty)$ in (2.7), we straddle the two extremes of *subspace closeness* and *data approximation*. In order to solve (2.7), we define an $m \times m$ symmetric matrix $\mathbf{A}_\ell = \sum_{p \neq \ell} \mathbf{D}_p\mathbf{D}_p^T + \frac{\lambda}{2}\widetilde{\mathbf{Y}}_\ell\widetilde{\mathbf{Y}}_\ell^T$. It then follows from [54] that (2.7) has a closed-form solution that involves eigendecomposition of $\mathbf{A}_\ell$. Specifically, $\mathbf{D}_\ell = \arg\max \operatorname{tr}(\mathbf{D}_\ell^T\mathbf{A}_\ell\mathbf{D}_\ell)$ is given by the first s eigenvectors of $\mathbf{A}_\ell$ associated with its s-largest eigenvalues.

This completes our description of the subspace update step. We can now combine the subspace assignment and subspace update steps to fully describe our algorithm for MC-UoS learning. This algorithm, which we term *metric-constrained union-of-subspaces learning* (MiCUSaL), is given by Algorithm 1. In terms of the complexity of this algorithm in each iteration, notice that the subspace assignment step requires $\mathcal{O}(mLsN)$ operations. In addition, the total number of operations needed to compute the $\mathbf{A}_\ell$'s in each iteration is $\mathcal{O}(m^2(Ls + N))$. Finally, each iteration also requires L eigendecompositions of $m \times m$ matrices, each one of which has $\mathcal{O}(m^3)$ complexity. Therefore, the computational complexity of MiCUSaL in each iteration is given by $\mathcal{O}(m^3L + m^2N + m^2Ls + mLsN)$. We conclude this discussion by pointing out that we

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

```
Input: Training data $\mathbf{Y} \in \mathbb{R}^{m \times N}$, number of subspaces $L$, dimension of
  subspaces $s$, and parameter $\lambda$.
Initialize: Random orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m \times s}\}_{\ell=1}^{L}$.
 1: $\bar{\mathbf{y}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i$, $\widetilde{\mathbf{y}}_i = \mathbf{y}_i - \bar{\mathbf{y}}$, $i = 1, \dots, N$.
 2: while stopping rule do
 3:   for $i = 1$ to $N$ (Subspace Assignment) do
 4:     $l_i = \arg\max_\ell \|\mathbf{D}_\ell^T \widetilde{\mathbf{y}}_i\|_2$.
 5:     $w_{l_i,i} = 1$ and $\forall \ell \neq l_i$, $w_{\ell,i} = 0$.
 6:   end for
 7:   for $\ell = 1$ to $L$ (Subspace Update) do
 8:     $\mathbf{c}_\ell = \{i \in \{1, \dots, N\} : w_{\ell,i} = 1\}$.
 9:     $\widetilde{\mathbf{Y}}_\ell = [\widetilde{\mathbf{y}}_i : i \in \mathbf{c}_\ell]$.
10:     $\mathbf{A}_\ell = \sum_{p \neq \ell} \mathbf{D}_p \mathbf{D}_p^T + \frac{\lambda}{2} \widetilde{\mathbf{Y}}_\ell \widetilde{\mathbf{Y}}_\ell^T$.
11:     Eigendecomposition of $\mathbf{A}_\ell = \mathbf{U}_\ell \mathbf{\Sigma}_\ell \mathbf{U}_\ell^T$.
12:     $\mathbf{D}_\ell =$ Columns of $\mathbf{U}_\ell$ corresponding to the $s$-largest diagonal elements in $\mathbf{\Sigma}_\ell$.
13:   end for
14: end while
Output: Orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m \times s}\}_{\ell=1}^{L}$.
```

cannot guarantee convergence of MiCUSaL to a global optimal solution. However, since the objective function F_1 in (2.5) is bounded below by zero and MiCUSaL ensures that F_1 does not increase after each iteration, it follows that MiCUSaL iterates do indeed converge (possibly to one of the local optimal solutions). This local convergence, of course, will be a function of the initialization of MiCUSaL. In this work, we advocate the use of random subspaces for initialization, while we study the impact of such random initialization in Section 2.5.

2.3 Practical Considerations

The MiCUSaL algorithm described in Section 2.2 requires knowledge of the number of subspaces L and the dimension of subspaces s. In practice, however, one cannot assume knowledge of these parameters a priori. Instead, one must estimate both the number and the dimension of subspaces from the training data themselves. In this section, we describe a generalization of the MiCUSaL algorithm that achieves this objective. Our algorithm, which we term *adaptive MC-UoS learning* (aMiCUSaL), requires only knowledge of loose upper bounds on L and s, which we denote by $L_{\max}$ and $s_{\max}$, respectively.

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

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

$$(\ell^*, p^*) = \arg\min_{\ell \neq p} d_u(\mathcal{S}_\ell, \mathcal{S}_p) \text{ s.t. } \frac{d_u(\mathcal{S}_{\ell^*}, \mathcal{S}_{p^*})}{\sqrt{s_{\max}}} \leq \epsilon_{\min}. \tag{2.8}$$

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

After subspace merging, we move onto the step of estimation of the dimension, s,

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

Algorithm 2: Adaptive MC-UoS Learning (aMiCUSaL)

```
Input: Training data $\mathbf{Y} \in \mathbb{R}^{m \times N}$, loose upper bounds $L_{\max}$ and $s_{\max}$, and
  parameters $\lambda$, $k_1$, $k_2$, $\epsilon_{\min}$.
Initialize: Random orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m \times s_{\max}}\}_{\ell=1}^{L_{\max}}$, and set $\widehat{L} = L_{\max}$.
 1: $\bar{\mathbf{y}} = \frac{1}{N}\sum_{i=1}^N \mathbf{y}_i$, $\widetilde{\mathbf{y}}_i = \mathbf{y}_i - \bar{\mathbf{y}}$, $i = 1, \dots, N$.
 2: while stopping rule do
 3:   Fix $\mathbf{D}$ and update $\mathbf{W}$ using (2.6). Also, set $\boldsymbol{\mathcal{T}} = \emptyset$ and $L_1 = 0$.
 4:   for $\ell = 1$ to $\widehat{L}$ (Subspace Pruning) do
 5:     $\mathbf{c}_\ell = \{i \in \{1, \dots, N\} : w_{\ell,i} = 1\}$.
 6:     If $|\mathbf{c}_\ell| \neq 0$ then $\mathbf{c}_{L_1+1} = \mathbf{c}_\ell$, $\widetilde{\mathbf{Y}}_{L_1+1} = [\widetilde{\mathbf{y}}_i : i \in \mathbf{c}_\ell]$, $L_1 = L_1 + 1$ and $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{T}} \cup \{\ell\}$.
 7:   end for
 8:   $\mathbf{D} = [\mathbf{D}_{\mathcal{T}_{(1)}}, \dots, \mathbf{D}_{\mathcal{T}_{(L_1)}}]$ and $\widehat{L} = L_1$.
 9:   Update each $\mathbf{D}_\ell$ ($\ell = 1, \dots, \widehat{L}$) in $\mathbf{D}$ using (2.7).
10: end while
11: $(\ell^*, p^*) = \arg\min_{\ell \neq p, \ell, p = 1, \dots, \widehat{L}} d_u(\mathcal{S}_\ell, \mathcal{S}_p)$.
12: while $\frac{d_u(\mathcal{S}_{\ell^*}, \mathcal{S}_{p^*})}{\sqrt{s_{\max}}} \leq \epsilon_{\min}$ (Subspace Merging) do
13:   Merge $\mathcal{S}_{\ell^*}$ and $\mathcal{S}_{p^*}$, and update $\mathbf{Y}_{\ell^*}$ and $\mathbf{D}_{\ell^*}$.
14:   $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_{\ell^*}, \dots, \mathbf{D}_{p^*-1}, \mathbf{D}_{p^*+1}, \dots, \mathbf{D}_{\widehat{L}}]$ and $\widehat{L} = \widehat{L} - 1$.
15:   $(\ell^*, p^*) = \arg\min_{\ell \neq p, \ell, p = 1, \dots, \widehat{L}} d_u(\mathcal{S}_\ell, \mathcal{S}_p)$.
16: end while
17: Fix $\mathbf{D}$, update $\mathbf{W} \in \mathbb{R}^{\widehat{L} \times N}$ using (2.6), and update $\{\mathbf{c}_\ell\}_{\ell=1}^{\widehat{L}}$.
18: for $\ell = 1$ to $\widehat{L}$ do
19:   $\widetilde{\mathbf{Y}}_\ell = [\widetilde{\mathbf{y}}_i : i \in \mathbf{c}_\ell]$ and $\widehat{\mathbf{Y}}_\ell = \mathbf{D}_\ell \mathbf{D}_\ell^T \widetilde{\mathbf{Y}}_\ell$.
20:   Calculate $\widehat{s}_\ell$ using (2.9) and (2.10).
21: end for
22: $s = \max\{\widehat{s}_1, \dots, \widehat{s}_{\widehat{L}}\}$.
23: $\widehat{\mathbf{D}}_\ell =$ First $s$ columns of $\mathbf{D}_\ell$, $\ell = 1, \dots, \widehat{L}$.
24: Initialize Algorithm 1 with $\{\widehat{\mathbf{D}}_\ell\}_{\ell=1}^{\widehat{L}}$ and update $\{\widehat{\mathbf{D}}_\ell\}_{\ell=1}^{\widehat{L}}$ using Algorithm 1.
Output: Orthonormal bases $\{\widehat{\mathbf{D}}_\ell \in \mathbb{R}^{m \times s}\}_{\ell=1}^{\widehat{L}}$.
```

of the subspaces. To this end, we first estimate the dimension of each subspace $\mathcal{S}_\ell$, denoted by s_ℓ, and then s is selected as the maximum of these s_ℓ's. There have been many efforts in the literature to estimate the dimension of a subspace; see, e.g., [55–58] for an incomplete list. In this work, we focus on the method given in [55], which formulates the maximum likelihood estimator (MLE) of s_ℓ. This is because: (*i*) the noise level is unknown in our problem, and (*ii*) the MLE in [55] has a simple form. However, the MLE of [55] is sensitive to noise. We therefore first apply a "smoothing" process before using that estimator. This involves first updating $\mathbf{W}$ (i.e., $\mathbf{c}_\ell$'s) using the updated $\mathbf{D}$ and "denoising" our data by projecting $\widetilde{\mathbf{Y}}_\ell$ onto $\mathcal{S}_\ell$, given by $\widehat{\mathbf{Y}}_\ell = \mathbf{D}_\ell \mathbf{D}_\ell^T \widetilde{\mathbf{Y}}_\ell$, and then using $\widehat{\mathbf{Y}}_\ell$ to estimate s_ℓ. For a given column $\widehat{\mathbf{y}}$ in $\widehat{\mathbf{Y}}_\ell$ and a fixed number of nearest neighbors k_0, the unbiased MLE of s_ℓ with respect to $\widehat{\mathbf{y}}$ is given by [55]

$$\widehat{s}_\ell^{k_0}(\widehat{\mathbf{y}}) = \Big[\frac{1}{k_0-2}\sum_{a=1}^{k_0-1}\log\frac{\Gamma_{k_0}(\widehat{\mathbf{y}})}{\Gamma_a(\widehat{\mathbf{y}})}\Big]^{-1}, \tag{2.9}$$

where $\Gamma_a(\widehat{\mathbf{y}})$ is the ℓ_2 distance between $\widehat{\mathbf{y}}$ and its a-th nearest neighbor in $\widehat{\mathbf{Y}}_\ell$. An estimate of s_ℓ can now be written as the average of all estimates with respect to every signal in $\widehat{\mathbf{Y}}_\ell$, i.e., $\widehat{s}_\ell^{k_0} = \frac{1}{|\mathbf{c}_\ell|}\sum_{i\in\mathbf{c}_\ell}\widehat{s}_\ell^{k_0}(\widehat{\mathbf{y}}_i)$. In fact, as suggested in [55], we estimate s_ℓ by averaging over a range of $k_0 = k_1,\dots,k_2$, i.e.,

$$\widehat{s}_\ell = \frac{1}{k_2-k_1+1}\sum_{k_0=k_1}^{k_2}\widehat{s}_\ell^{k_0}. \tag{2.10}$$

Once we get an estimate $s = \max_\ell \widehat{s}_\ell$, we trim each orthonormal basis by keeping only the first s columns of each (ordered) orthonormal basis $\mathbf{D}_\ell$ in our collection, which is denoted by $\widehat{\mathbf{D}}_\ell$.[4] Given the bases $\{\widehat{\mathbf{D}}_\ell \in \mathbb{R}^{m\times s}\}_{\ell=1}^{\widehat{L}}$, we finally run MiCUSaL again that is initialized using these $\widehat{\mathbf{D}}_\ell$'s until it converges. Combining all the steps mentioned above, we can now formally describe *adaptive MC-UoS learning* (aMiCUSaL) in Algorithm 2.

2.4 MC-UoS Learning Using Missing Data

In this section, we study MC-UoS learning for the case of training data having missing entries. To be specific, for each $\mathbf{y}_i$ in $\mathbf{Y}$, we assume to only observe its entries at locations given by the set $\mathbf{\Omega}_i \subset \{1,\dots,m\}$ with $|\mathbf{\Omega}_i| > s$, which is denoted by $[\mathbf{y}_i]_{\mathbf{\Omega}_i} \in \mathbb{R}^{|\mathbf{\Omega}_i|}$. Since we do not have access to the complete data, it is impossible to compute the quantities $\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ in (2.2) explicitly. But, it is shown in [59] that $\|[\mathbf{y}_i]_{\mathbf{\Omega}_i} - P_{\mathcal{S}_{\ell\mathbf{\Omega}_i}}[\mathbf{y}_i]_{\mathbf{\Omega}_i}\|_2^2$ for uniformly random $\mathbf{\Omega}_i$ is very close to $\frac{|\mathbf{\Omega}_i|}{m}\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ with very high probability as long as $|\mathbf{\Omega}_i|$ is slightly greater than s. Here, $P_{\mathcal{S}_{\ell\mathbf{\Omega}_i}}$ is defined as $P_{\mathcal{S}_{\ell\mathbf{\Omega}_i}} = [\mathbf{D}_\ell]_{\mathbf{\Omega}_i,:}([\mathbf{D}_\ell]_{\mathbf{\Omega}_i,:})^\dagger$ with $([\mathbf{D}_\ell]_{\mathbf{\Omega}_i,:})^\dagger = \big([\mathbf{D}_\ell]_{\mathbf{\Omega}_i,:}^T[\mathbf{D}_\ell]_{\mathbf{\Omega}_i,:}\big)^{-1}[\mathbf{D}_\ell]_{\mathbf{\Omega}_i,:}^T$. Motivated by this, we replace $\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ by $\frac{m}{|\mathbf{\Omega}_i|}\|[\mathbf{y}_i]_{\mathbf{\Omega}_i} - P_{\mathcal{S}_{\ell\mathbf{\Omega}_i}}[\mathbf{y}_i]_{\mathbf{\Omega}_i}\|_2^2$ in (2.2) and reformulate

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

the MC-UoS learning problem as $(\mathbf{D},\mathbf{W}) = \arg\min_{\mathbf{D},\mathbf{W}} F_2(\mathbf{D},\mathbf{W})$ with the objective function $F_2(\mathbf{D},\mathbf{W})$ given by

$$F_2(\mathbf{D},\mathbf{W}) = \sum_{\substack{\ell,p=1 \\ \ell\neq p}}^{L} \|\mathbf{D}_\ell - P_{\mathcal{S}_p}\mathbf{D}_\ell\|_F^2 + \lambda \sum_{i=1}^{N}\sum_{\ell=1}^{L} w_{\ell,i}\frac{m}{|\mathbf{\Omega}_i|}\big\|[\mathbf{y}_i]_{\mathbf{\Omega}_i} - P_{\mathcal{S}_{\ell\mathbf{\Omega}_i}}[\mathbf{y}_i]_{\mathbf{\Omega}_i}\big\|_2^2. \quad (2.11)$$

As in Section 2.2, we propose to solve this problem by making use of alternating minimization that comprises subspace assignment and subspace update steps. To this end, we again initialize $\mathbf{D}$ such that each block $\mathbf{D}_\ell \in \mathbb{R}^{m\times s}$ is a random orthonormal basis. Next, when $\mathbf{D}$ is fixed, subspace assignment corresponds to solving for each $i = 1,\dots,N$,

$$l_i = \operatorname*{arg\,min}_{\ell=1,\dots,L} \|[\mathbf{y}_i]_{\mathbf{\Omega}_i} - P_{\mathcal{S}_{\ell\mathbf{\Omega}_i}}[\mathbf{y}_i]_{\mathbf{\Omega}_i}\|_2^2, \quad (2.12)$$

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

$$f_2(\mathbf{D}_\ell) = -\mathrm{tr}(\mathbf{D}_\ell^T\mathbf{A}_\ell\mathbf{D}_\ell) + \frac{\lambda}{2}\sum_{i\in\mathbf{c}_\ell}\frac{m}{|\mathbf{\Omega}_i|}\big\|[\mathbf{y}_i]_{\mathbf{\Omega}_i} - P_{\mathcal{S}_{\ell\mathbf{\Omega}_i}}[\mathbf{y}_i]_{\mathbf{\Omega}_i}\big\|_2^2. \quad (2.13)$$

Here, $\mathbf{c}_\ell$ is as defined in Section 2.2 and $\mathbf{A}_\ell = \sum_{p\neq\ell}\mathbf{D}_p\mathbf{D}_p^T$. It is also easy to verify that $f_2(\mathbf{D}_\ell)$ is invariant to the choice of the orthonormal basis of $\mathcal{S}_\ell$; hence, we can treat $\min_{\mathbf{D}_\ell\in\mathcal{V}_{m,s}} f_2(\mathbf{D}_\ell)$ as an optimization problem on the Grassmann manifold [60]. Note that we can rewrite $f_2(\mathbf{D}_\ell)$ as $f_2(\mathbf{D}_\ell) = \sum_{q=0}^{|\mathbf{c}_\ell|} f_2^{(q)}(\mathbf{D}_\ell)$, where $f_2^{(0)}(\mathbf{D}_\ell) = -\mathrm{tr}(\mathbf{D}_\ell^T\mathbf{A}_\ell\mathbf{D}_\ell)$ and $f_2^{(q)}(\mathbf{D}_\ell) = \frac{\lambda m}{2|\mathbf{\Omega}_{c_{\ell(q)}}|}\|[\mathbf{y}_{c_{\ell(q)}}]_{\mathbf{\Omega}_{c_{\ell(q)}}} - P_{\mathcal{S}_{\ell\mathbf{\Omega}_{c_{\ell(q)}}}}[\mathbf{y}_{c_{\ell(q)}}]_{\mathbf{\Omega}_{c_{\ell(q)}}}\|_2^2$ for $q = 1,\dots,|\mathbf{c}_\ell|$. In here, $c_{\ell(q)}$ denotes the q-th element in $\mathbf{c}_\ell$. In order to minimize $f_2(\mathbf{D}_\ell)$, we employ incremental gradient descent procedure on Grassmann manifold [61], which performs the update with respect to a single component of $f_2(\mathbf{D}_\ell)$ in each step. To be specific, we first compute the gradient of a single cost function $f_2^{(q)}(\mathbf{D}_\ell)$ in $f_2(\mathbf{D}_\ell)$, and move along a short geodesic curve in the gradient direction. For instance, the gradient of $f_2^{(0)}(\mathbf{D}_\ell)$ is

$$\nabla f_2^{(0)} = (\mathbf{I}_m - \mathbf{D}_\ell\mathbf{D}_\ell^T)\frac{df_2^{(0)}}{d\mathbf{D}_\ell} = -2(\mathbf{I}_m - \mathbf{D}_\ell\mathbf{D}_\ell^T)\mathbf{A}_\ell\mathbf{D}_\ell. \quad (2.14)$$

Then the geodesic equation emanating from $\mathbf{D}_\ell$ in the direction $-\nabla f_2^{(0)}$ with a step length η is given by [60]

$$\mathbf{D}_\ell(\eta) = \mathbf{D}_\ell \mathbf{V}_\ell \cos(\mathbf{\Sigma}_\ell \eta)\mathbf{V}_\ell^T + \mathbf{U}_\ell \sin(\mathbf{\Sigma}_\ell \eta)\mathbf{V}_\ell^T, \tag{2.15}$$

where $\mathbf{U}_\ell \mathbf{\Sigma}_\ell \mathbf{V}_\ell^T$ is the SVD decomposition of $-\nabla f_2^{(0)}$. The update of $\mathbf{D}_\ell$ with respect to $f_2^{(q)}(\mathbf{D}_\ell)$ $(q = 1, \dots, |\mathbf{c}_\ell|)$ can be performed as in the GROUSE algorithm [62] *but* with a step size $\frac{\lambda m}{2|\mathbf{\Omega}_{c_{\ell(q)}}|}\eta$. In order for f_2 to converge, we also reduce the step size after each iteration [62]. We complete this discussion by presenting our learning algorithm for missing data in Algorithm 3, termed *robust MC-UoS learning* (rMiCUSaL). By defining $T = \max_i |\mathbf{\Omega}_i|$, the subspace assignment step of rMiCUSaL requires $\mathcal{O}(TLs^2N)$ flops in each iteration [62]. Computing the $\mathbf{A}_\ell$'s in each iteration requires $\mathcal{O}(m^2Ls)$ operations. Next, the cost of updating each $\mathbf{D}_\ell$ with respect to $f_2^{(0)}(\mathbf{D}_\ell)$ is $\mathcal{O}(m^3)$, while it is $\mathcal{O}(ms + Ts^2)$ with respect to $f_2^{(q)}(\mathbf{D}_\ell)$ for $q \neq 0$ [62]. It therefore follows that the computational complexity of rMiCUSaL in each iteration is $\mathcal{O}(m^3L + m^2Ls + msN + TLs^2N)$. We refer the reader to Section 2.5 for exact running time of rMiCUSaL on training data.

2.5 Experimental Results

In this section, we examine the effectiveness of MC-UoS learning using Algorithms 1–3. For the complete data experiments, we compare MiCUSaL/aMiCUSaL with several state-of-the-art UoS learning algorithms such as Block-Sparse Dictionary Design (SAC+BK-SVD) [30], K-subspace clustering (K-sub) [53], Sparse Subspace Clustering (SSC) [13], Robust Sparse Subspace Clustering (RSSC) [34], Robust Subspace Clustering via Thresholding (TSC) [33], as well as with Principal Component Analysis (PCA) [1]. In the case of the UoS learning algorithms, we use codes provided by their respective authors. In the case of SSC, we use the noisy variant of the optimization program in [13] and set $\lambda_z = \alpha_z/\mu_z$ in all experiments, where λ_z and μ_z are as defined in [13, (13) & (14)], while the parameter α_z varies in different experiments. In the case of RSSC, we set $\lambda = 1/\sqrt{s}$ as per [34], while the tuning parameter in TSC is set $q = \max(3, \lceil N/(L \times 20) \rceil)$ when L is provided. For the case of training data having missing entries, we compare the results of rMiCUSaL with k-GROUSE [22] and

Algorithm 3: Robust MC-UoS learning (rMiCUSaL)

```
Input: Training data $\{[\mathbf{y}_i]_{\mathbf{\Omega}_i}\}_{i=1}^N$, number of subspaces $L$, dimension of
  subspaces $s$, and parameters $\lambda$ and $\eta$.
Initialize: Random orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m \times s}\}_{\ell=1}^L$.
1: while stopping rule do
2:   for $i = 1$ to $N$ (Subspace Assignment) do
3:     $l_i = \arg\min_\ell \|[\mathbf{y}_i]_{\mathbf{\Omega}_i} - P_{\mathcal{S}_{\ell \mathbf{\Omega}_i}}[\mathbf{y}_i]_{\mathbf{\Omega}_i}\|_2^2$.
4:     $w_{l_i,i} = 1$ and $\forall \ell \neq l_i$, $w_{\ell,i} = 0$.
5:   end for
6:   for $\ell = 1$ to $L$ (Subspace Update) do
7:     $\mathbf{c}_\ell = \{i \in \{1, \dots, N\} : w_{\ell,i} = 1\}$, $t = 0$.
8:     while stopping rule do
9:       $t = t + 1$, $\eta_t = \frac{\eta}{t}$.
10:      $\mathbf{A}_\ell = \sum_{p \neq \ell} \mathbf{D}_p \mathbf{D}_p^T$, $\mathbf{\Delta}_\ell = 2(\mathbf{I}_m - \mathbf{D}_\ell \mathbf{D}_\ell^T)\mathbf{A}_\ell \mathbf{D}_\ell$.
11:      $\mathbf{D}_\ell = \mathbf{D}_\ell \mathbf{V}_\ell \cos(\mathbf{\Sigma}_\ell \eta_t)\mathbf{V}_\ell^T + \mathbf{U}_\ell \sin(\mathbf{\Sigma}_\ell \eta_t)\mathbf{V}_\ell^T$, where $\mathbf{U}_\ell \mathbf{\Sigma}_\ell \mathbf{V}_\ell^T$ is the compact
         SVD of $\mathbf{\Delta}_\ell$.
12:      for $q = 1$ to $|\mathbf{c}_\ell|$ do
13:        $\boldsymbol{\theta} = ([\mathbf{D}_\ell]_{\mathbf{\Omega}_{c_{\ell(q)}},:})^\dagger [\mathbf{y}_{c_{\ell(q)}}]_{\mathbf{\Omega}_{c_{\ell(q)}}}$, $\boldsymbol{\omega} = \mathbf{D}_\ell \boldsymbol{\theta}$.
14:        $\mathbf{r} = \mathbf{0}_m$, $[\mathbf{r}]_{\mathbf{\Omega}_{c_{\ell(q)}}} = [\mathbf{y}_{c_{\ell(q)}}]_{\mathbf{\Omega}_{c_{\ell(q)}}} - [\boldsymbol{\omega}]_{\mathbf{\Omega}_{c_{\ell(q)}}}$.
15:        $\mathbf{D}_\ell = \mathbf{D}_\ell + \Big((\cos(\mu \frac{\lambda m}{|\mathbf{\Omega}_{c_{\ell(q)}}|}\eta_t) - 1)\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|_2} + \sin(\mu \frac{\lambda m}{|\mathbf{\Omega}_{c_{\ell(q)}}|}\eta_t)\frac{\mathbf{r}}{\|\mathbf{r}\|_2}\Big)\frac{\boldsymbol{\theta}^T}{\|\boldsymbol{\theta}\|_2}$, where
           $\mu = \|\mathbf{r}\|_2 \|\boldsymbol{\omega}\|_2$.
16:      end for
17:    end while
18:  end for
19: end while
Output: Orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m \times s}\}_{\ell=1}^L$.
```

GROUSE [62].[5]

In order to generate noisy training and test data in these experiments, we start with sets of "clean" training and test samples, denoted by $\mathbf{X}$ and $\mathbf{X}^{te}$, respectively. We then add white Gaussian noise to these samples to generate noisy training and test samples $\mathbf{Y}$ and $\mathbf{Z}$, respectively. In the following, we use σ_{tr}^2 and σ_{te}^2 to denote variance of noise added to training and test samples, respectively. In the missing data experiments, for every fixed noise variance σ_{tr}^2, we create training (but not test) data with different percentages of missing values, where the number of missing entries is set to be 10%, 30% and 50% of the signal dimension. Our reported results are based

[5]As discussed in [63], we omit the results for SSC with missing data in this thesis because it fills in the missing entries with random values, resulting in relatively poor performance for problems with missing data.

on random initializations of MiCUSaL and aMiCUSaL algorithms. In this regard, we adopt the following simple approach to mitigate any stability issues that might arise due to random initializations. We perform multiple random initializations for every fixed $\mathbf{Y}$ and λ, and then retain the learned MC-UoS structure that results in the smallest value of the objective function in (2.2). We also use a similar approach for selecting the final structures returned by K-subspace clustering and Block-Sparse Dictionary Design, with the only difference being that (2.2) in this case is replaced by the approximation error of training data.

2.5.1 Experiments on Synthetic Data

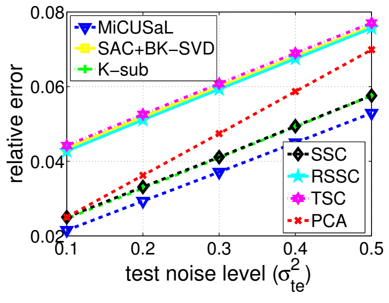
In the first set of synthetic experiments, we consider $L = 5$ subspaces of the same dimension $s = 13$ in an $m = 180$-dimensional ambient space. The five subspaces $\mathcal{S}_\ell$'s of $\mathbb{R}^{180}$ are defined by their orthonormal bases $\{\mathbf{T}_\ell \in \mathbb{R}^{m \times s}\}_{\ell=1}^{5}$ as follows. We start with a random orthonormal basis $\mathbf{T}_1 \in \mathbb{R}^{m \times s}$ and for every $\ell \geq 2$, we set $\mathbf{T}_\ell = \text{orth}(\mathbf{T}_{\ell-1} + t_s \mathbf{W}_\ell)$ where every entry in $\mathbf{W}_\ell \in \mathbb{R}^{m \times s}$ is a uniformly distributed random number between 0 and 1, and $\text{orth}(\cdot)$ 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_ℓ points from $\mathcal{S}_\ell$ as $\mathbf{X}_\ell = \mathbf{T}_\ell \mathbf{C}_\ell$, where $\mathbf{C}_\ell \in \mathbb{R}^{s \times n_\ell}$ is a matrix whose elements are drawn independently and identically from $\mathcal{N}(0, 1)$ distribution. In here, we set $n_1 = n_3 = n_5 = 150$, and $n_2 = n_4 = 100$; hence, $N = 650$. We then stack all the data into a matrix $\mathbf{X} = [\mathbf{X}_1, \dots, \mathbf{X}_5] = \{\mathbf{x}_i\}_{i=1}^{N}$ and normalize all the samples to unit ℓ_2 norms. Test data $\mathbf{X}^{te} \in \mathbb{R}^{m \times N}$ are produced using the same foregoing strategy. Then we add white Gaussian noise with different expected noise power to both $\mathbf{X}$ and $\mathbf{X}^{te}$. Specifically, we set σ_{tr}^2 to be 0.1, while σ_{te}^2 ranges from 0.1 to 0.5. We generate $\mathbf{X}$ and $\mathbf{X}^{te}$ 10 times, while Monte Carlo simulations for noisy data are repeated 20 times for every fixed $\mathbf{X}$ and $\mathbf{X}^{te}$. Therefore, the results reported in here correspond to an average of 200 random trials.

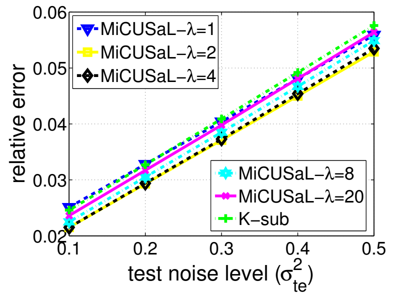
We make use of the collection of noisy samples, $\mathbf{Y}$, to learn a union of L subspaces of dimension s and stack the learned orthonormal bases $\{\mathbf{D}_\ell\}_{\ell=1}^{L}$ into $\mathbf{D}$. In this set

Table 2.1: d_{avg} of different UoS learning algorithms for synthetic data

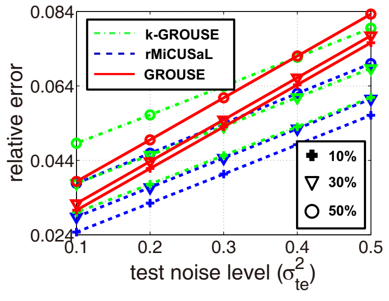
d_{avg}	Algorithms		
Complete	MiCUSaL($\lambda = 2$)	SAC+BK-SVD	K-sub
	0.1331	0.2187	0.1612
	SSC	RSSC	TSC
	0.1386	0.2215	0.2275
Missing	rMiCUSaL-10%	rMiCUSaL-30%	rMiCUSaL-50%
	0.1661	**0.1788**	**0.2047**
	k-GROUSE-10%	k-GROUSE-30%	k-GROUSE-50%
	0.3836	0.4168	0.4649



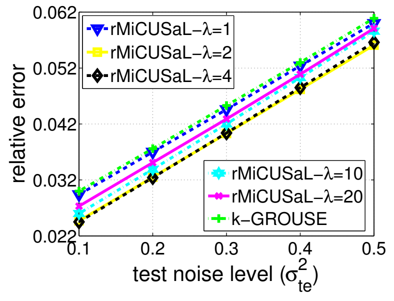
(a)



(b)



(c)



(d)

Figure 2.2: Comparison of MC-UoS learning performance on synthetic data. (a) and (c) show relative errors of test signals for complete and missing data experiments, where $\lambda = 2$ for both MiCUSaL and rMiCUSaL. The numbers in the legend of (c) indicate the percentages of missing entries within the training data. (b) and (d) show relative errors of test signals for MiCUSaL and rMiCUSaL (with 10% missing entries) using different λ's.

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

To compare with other UoS learning methods, we choose $\lambda = 2$ for both MiCUSaL and rMiCUSaL. In the complete data experiments, we perform SSC with $\alpha_z = 60$. We set the subspace dimension for PCA to be the (unrealizable, ideal) one that yields the best denoising result on *training* samples. We also use the same subspace dimension (again, unrealizable and ideal) for GROUSE in the corresponding missing data experiments. Table 2.1 summarizes the d_{avg}'s of different UoS learning algorithms for both complete and missing data experiments. As can be seen, MiCUSaL produces smaller d_{avg}'s, which in turn leads to smaller relative errors of test data; see Fig. 2.2(a) for a validation of this claim. For MC-UoS learning with missing data, rMiCUSaL also learns a better MC-UoS in that: (i) for a fixed percentage of the number of missing observations in the training data, the d_{avg} for rMiCUSaL is much smaller than the one for k-GROUSE (see Table 2.1); and (*ii*) rMiCUSaL outperforms k-GROUSE and GROUSE in terms of smaller reconstruction errors of test data. Moreover, we can infer from Fig. 2.2(c) that for a fixed σ_{te}, when the number of missing entries increases, the performance of rMiCUSaL degrades less compared to k-GROUSE. We also test the UoS learning performance with complete data when the subspaces are not close to each other (e.g., $t_s = 0.2$). In this case, all the UoS learning algorithms, including MiCUSaL, learn the subspaces successfully. We omit these plots because of space constraints.

Table 2.2: d_{avg} of MiCUSaL and rMiCUSaL for different λ's using synthetic data

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

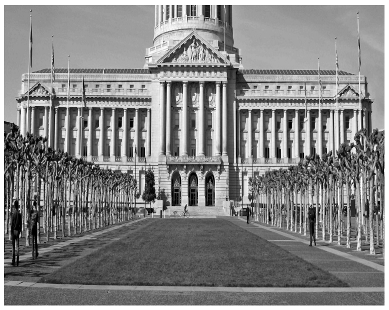
For both MiCUSaL and rMiCUSaL, we also analyze the effect of the key parameter, λ, on the UoS learning performance. We implement MiCUSaL with $\lambda \in \{1, 2, 4, 8, 20\}$ in the complete data experiments and select $\lambda \in \{1, 2, 4, 10, 20\}$ for rMiCUSaL in the missing data experiments, where the number of missing entries in the training data is 10% of the signal dimension. The results are shown in Fig. 2.2(b), Fig. 2.2(d) and Table 2.2. We can see when $\lambda = 1$, both the d_{avg}'s and reconstruction errors of the test data are large for MiCUSaL and rMiCUSaL. This is because the learned subspaces are too close to each other, which results in poor data representation capability of the learned $\mathbf{D}_\ell$'s. When $\lambda = 2$ or 4, both these algorithms achieve good performance in terms of small d_{avg}'s and relative errors of test data. As λ increases further, both d_{avg} and relative errors of test data also increase. Furthermore, as λ grows, the curves of relative errors of test data for MiCUSaL and rMiCUSaL get closer to the ones for K-sub and k-GROUSE, respectively. This phenomenon coincides with our discussion in Section 2.2 and 2.4. Finally, we note that both MiCUSaL and rMiCUSaL achieve their best performance when $\lambda \in [2, 4]$, and deviations of the representation errors of test data are very small when λ falls in this range.

Next, we study the effect of random initialization of subspaces on MiCUSaL performance by calculating the standard deviation of the mean of the reconstruction errors of the test data for the 8 random initializations. The mean of these 200 standard deviations ends up being only 0.003 for all σ_{te}'s when $\lambda = 2$. In addition, as λ gets larger, the variation of the results increases only slightly (the mean of the standard deviations is 0.0034 for $\lambda = 8$). On the other hand, the mean of the standard deviations for K-sub is 0.0039. Furthermore, the performance gaps between MiCUSaL and all other methods are larger than 0.003. Finally, the learned MC-UoS structure that results in the smallest value of the objective function (2.2) always results in the best denoising performance. This suggests that MiCUSaL always generates the best results and it is mostly insensitive to the choice of initial subspaces during the random initialization.

Table 2.3: Running time comparison (in sec) for rMiCUSaL and k-GROUSE

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

We also examine the running times of rMiCUSaL and k-GROUSE per iteration, which include both the subspace assignment and subspace update stages. For each subspace $\mathcal{S}_\ell$, we implement the optimization over $\mathbf{D}_\ell$ (i.e., Steps 8 to 17 in Algorithm 3) for 100 iterations. All experiments are carried out using Matlab R2013a on an Intel i7-2600 3.4GHz CPU with 16 GB RAM. From the fourth row of Table 2.3, we observe rMiCUSaL takes slightly more time compared to k-GROUSE because rMiCUSaL needs two more steps for updating $\mathbf{D}_\ell$ (Steps 10 and 11 in Algorithm 3). However, the advantage of rMiCUSaL over k-GROUSE in learning a better UoS significantly outweighs this slight increase in computational complexity. We can also see that as the number of missing entries increases, both algorithms become faster. The reason for this is that when $|\mathbf{\Omega}_i|$ decreases for all i's, less time is needed during the subspace assignment step and for computing $\boldsymbol{\theta}$ and $\mathbf{r}$ in Algorithm 3.


(a)

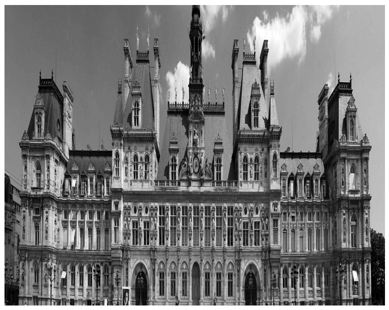
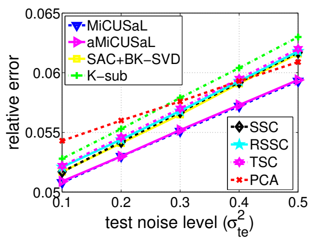

(b)

Figure 2.3: (a) San Francisco City Hall image. (b) Paris City Hall image.

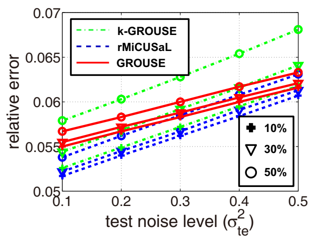
2.5.2 Experiments on City Scene Data

To further show the effectiveness of the proposed approaches, we test our proposed methods on real-world city scene data. First, we study the performance of our methods on San Francisco City Hall image, as shown in Fig. 2.3(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 30×20 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 ℓ_2 norms and are then used as signals in $\mathbf{X}$. Test signals in $\mathbf{X}^{te} \in \mathbb{R}^{600\times 722}$ are extracted in the same way from the right subimage. White Gaussian noise is then added to $\mathbf{X}$ and $\mathbf{X}^{te}$ separately, forming $\mathbf{Y}$ and $\mathbf{Z}$, respectively. In these experiments, σ_{tr}^2 is set to be 0.02 and 0.05, while σ_{te}^2 again ranges from 0.1 to 0.5. The Monte Carlo simulations for noisy data are repeated 50 times and the results reported here correspond to the average of these 50 trials. Note that each patch is treated as a single signal here, and our goal is to learn an MC-UoS from $\mathbf{Y}$ such that every test patch can be reliably denoised using the learned subspaces.

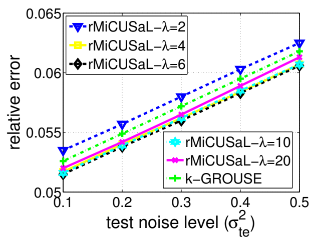
We perform aMiCUSaL on the training data $\mathbf{Y}$ with parameters $L_{\max} = 8$, $s_{\max} = 20$, $\lambda = 4$, $k_1 = 6$, $k_2 = 10$ and $\epsilon_{\min} = 0.08$. The number of random initializations that are used to arrive at the final MC-UoS structure using aMiCUSaL is 10 for every fixed $\mathbf{Y}$. The output L from aMiCUSaL is 4 or 5 and s is always between 11 and 13. We also perform MiCUSaL with the same L and s 10 times. For fair comparison, we also use the method in this thesis to get the dimension of the subspace for PCA, in



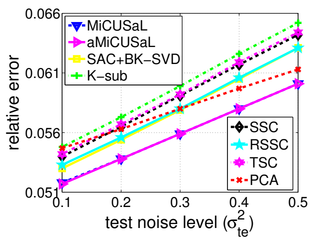
(a) $\sigma_{tr}^2 = 0.02$



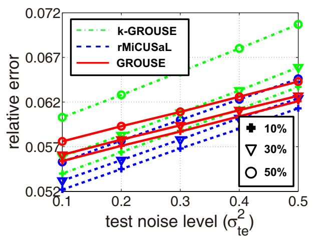
(b) $\sigma_{tr}^2 = 0.02$



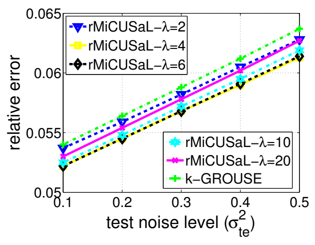
(c) $\sigma_{tr}^2 = 0.02$



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



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



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

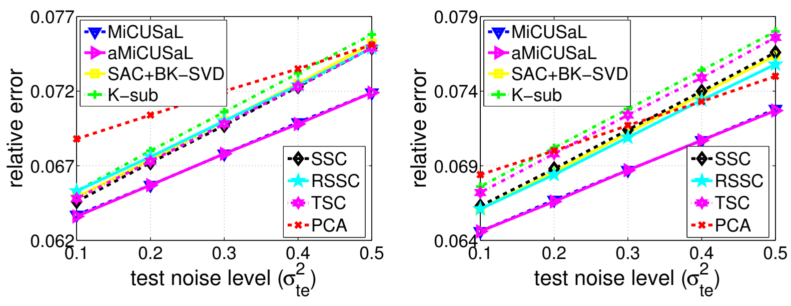
Figure 2.4: Comparison of MC-UoS learning performance on San Francisco City Hall data. (a) and (d) show relative errors of test signals for complete data experiments. (b) and (e) show relative errors of test signals for missing data experiments. The numbers in the legend of (b) and (e) indicate the percentages of missing entries within the training data. (c) and (f) show relative errors of test signals for rMiCUSaL (with 10% missing entries) using different λ's.

which case the estimated s is always 10. Note that for all state-of-the-art UoS learning algorithms, we use the same L and s as aMiCUSaL instead of using the L generated by the algorithms themselves. The reason for this is as follows. The returned L by SSC (with $\alpha_z = 40$) is 1. Therefore SSC reduces to PCA in this setting. The output L for RSSC is also 4 or 5, which coincides with our algorithm. The estimation of L (with $q = 2\max(3, \lceil N/(L \times 20)\rceil))$ for TSC is sensitive to the noise and data. Specifically, the estimated L is always from 6 to 9 for $\sigma_{tr}^2 = 0.02$ and L is always 1 when $\sigma_{tr}^2 = 0.05$, which results in poorer performance compared to the case when $L = 4$ or 5 for both training noise levels. In the missing data experiments, we set $L = 5$ and $s = 12$ for rMiCUSaL (with $\lambda = 4$) and k-GROUSE, and $s = 10$ for GROUSE. Fig. 2.4(a) and Fig. 2.4(d) describe the relative reconstruction errors of test samples when the training data are complete. We see both MiCUSaL and aMiCUSaL learn a better MC-UoS since they give rise to smaller relative errors of test data. Further, the average standard deviation of the mean of relative errors for test data is around 0.00015 for MiCUSaL and

0.00045 for K-sub. It can be inferred from Fig. 2.4(b) and Fig. 2.4(e) that rMiCUSaL also yields better data representation performance for the missing data case.

To examine the effect of λ on the denoising result in both complete and missing data experiments, we first run aMiCUSaL with $\lambda \in \{1, 2, 4, 6, 8, 10\}$ without changing other parameters. When $\lambda = 1$ or 2, aMiCUSaL always returns $L = 2$ or 3 subspaces, but the reconstruction errors of the test data are slightly larger than those for $\lambda = 4$. When $\lambda \geq 6$, the distances between the learned subspaces become larger, and the resulting L will be at least 6 when $\epsilon_{\min}$ is fixed. However, the relative errors of test data are still very close to the ones for $\lambda = 4$. This suggests that $\lambda = 4$ is a good choice in this setting since it leads to the smallest number of subspaces L and the best representation performance. We also perform rMiCUSaL with $\lambda \in \{2, 4, 6, 10, 20\}$ while keeping L and s fixed, where the number of missing entries in the training data is again 10% of the signal dimension. We show the relative errors of test data in Fig. 2.4(c) and Fig. 2.4(f). Similar to the results of the experiments with synthetic data, we again observe the fact that when λ is small (e.g., $\lambda = 2$), the reconstruction errors of the test data are large because the subspace closeness metric dominates in learning the UoS. The results for $\lambda = 4$ and 6 are very similar. As λ increases further, the performance of rMiCUSaL gets closer to that of k-GROUSE. We again report the running time of rMiCUSaL and k-GROUSE per iteration in the seventh row of Table 2.3, where we perform the optimization over each $\mathbf{D}_\ell$ for 100 iterations in each subspace update step for both rMiCUSaL and k-GROUSE. In these experiments, rMiCUSaL appears much slower than k-GROUSE. However, as presented in Fig. 2.4(b) and Fig. 2.4(e), the performance of rMiCUSaL is significantly better than k-GROUSE.

Next, we repeat these experiments for the complete data experiments using Paris City Hall image in Fig. 2.3(b), forming $\mathbf{X}, \mathbf{X}^{te} \in \mathbb{R}^{600 \times 950}$. We perform aMiCUSaL using the same parameters ($\lambda = 4$) as in the previous experiments. The estimated L in this case is always between 5 and 6 and s is always between 11 and 12. The estimated dimension of the subspace in PCA is 9 or 10 when $\sigma_{tr}^2 = 0.02$ and it is always 10 when $\sigma_{tr}^2 = 0.05$. In these experiments, we again use the same L and s as aMiCUSaL for all state-of-the-art UoS learning algorithms. This is because the returned L by SSC (with



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

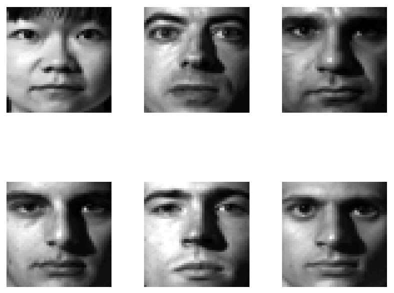
Figure 2.5: Comparison of MC-UoS learning performance on Paris City Hall data when the training data are complete.

$\alpha_z = 20$) is again 1 in this case. The estimated L by RSSC is usually 7 or 8, and the reconstruction errors of test data are very close to the ones reported here. If we apply TSC using the L estimated by itself (again, with $q = 2\max(3, \lceil N/(L \times 20)\rceil)$), we will have $L = 4$ when $\sigma^2_{tr} = 0.02$, while the relative errors of test data are very close to the results shown here. For $\sigma^2_{tr} = 0.05$, TSC will again result in only one subspace. The relative reconstruction errors of test data with different training noise levels are shown in Fig. 2.5, from which we make the conclusion that our methods obtain small errors, thereby outperforming all other algorithms. The average standard deviation of the mean of relative errors for test data is also smaller for MiCUSaL (around 0.00023) compared to K-sub (around 0.00037).

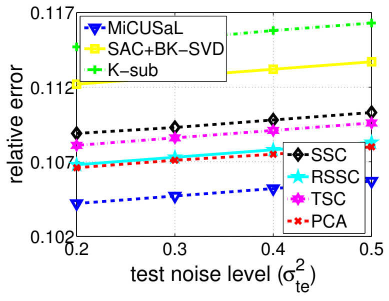
2.5.3 Experiments on Face Dataset

In this section, we work with the Extended Yale B dataset [49], which contains a set of 192×168 cropped images of 38 subjects. For each individual, there are 64 images taken under varying illumination conditions. We downsample the images to 48×42 pixels and each image is vectorized and treated as a signal; thus, $m = 2016$. It has been shown in [50] that the set of images of a given subject taken under varying illumination conditions can be well represented by a 9-dimensional subspace.

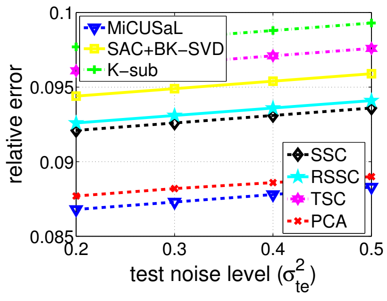
We first focus on a collection of images of subjects 5, 6 and 8 and normalize all the images to have unit ℓ_2 norms. Some representative images are presented in the first row of Fig. 2.6(a). Here we assume the images of these three subjects lie close to an MC-UoS



(a)



(b)



(c)

Figure 2.6: Comparison of MC-UoS learning performance on Extended Yale B dataset. The first row of (a) shows some images of subject 5, 6, 8 and the second row presents some images of subject 22, 28, 30. (b) and (c) show relative errors of test data in the two experiments.

with $L = 3$ and $s = 9$. For each set of images from one subject, we randomly select half of them for training and the remaining 32 images belong to test samples; therefore, $\mathbf{X}, \mathbf{X}^{te} \in \mathbb{R}^{2016\times 96}$. Then we add white Gaussian noise to both $\mathbf{X}$ and $\mathbf{X}^{te}$ and obtain $\mathbf{Y}$ and $\mathbf{Z}$. The random selection for generating $\mathbf{X}$ and $\mathbf{X}^{te}$ is repeated 10 times and we conduct Monte Carlo simulations for noisy data 10 times for every fixed $\mathbf{X}$ and $\mathbf{X}^{te}$. In these experiments, the value σ^2_{tr} is equal to 0.2 and σ^2_{te} is from 0.2 to 0.5. For fair comparison, the dimension of the subspace for PCA is set to be 9. We apply MiCUSaL with parameter $\lambda = 2$ and SSC with $\alpha_z = 40$. The number of random initializations in these experiments for both MiCUSaL and K-sub is set at 8 for every fixed $\mathbf{Y}$. Once again, we observe that MiCUSaL outperforms other learning methods, since it results in smaller relative errors (cf. Fig. 2.6(b)). Moreover, the average standard deviation of MiCUSaL for the 100 realizations of $\mathbf{Y}$ and $\mathbf{Z}$ is only 0.0013 for all σ_{te}'s, which is again smaller than that of K-sub (the corresponding value is 0.002).

We then repeat these experiments using a set of images of subjects 22, 28 and 30, and show some image samples in the second row of Fig. 2.6(a). We set $L = 3$, $s = 9$ and $\lambda = 2$ for MiCUSaL and $\alpha_z = 40$ for SSC. We again provide evidence in Fig. 2.6(c) that MiCUSaL yields better data representation performance in this setting. The average standard deviation of the mean of the reconstruction errors for test data is around 0.0012 for MiCUSaL and 0.0019 for K-sub in this case.

Chapter 3

Metric-Constrained Kernel Union-of-Subspaces

3.1 Problem Formulation

In this chapter, we study the geometry learning problem corresponds to the case of high-dimensional data drawn from a mixture of nonlinear manifolds in the ambient space $\mathbb{R}^m$. The basic premise in this case is that when data drawn from a mixture of nonlinear manifolds are mapped through a nonlinear map $\phi : \mathbb{R}^m \to \mathcal{F}$ to a higher-dimensional feature space $\mathcal{F} \subset \mathbb{R}^{\widetilde{m}}$ with $\widetilde{m} \ggg m$, then the ϕ-mapped "images" of these data can be modeled as lying near an MC-UoS $\mathcal{M}_L$ in the feature space. In order to learn this model, we once again assume access to a collection of N training samples, $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N] \in \mathbb{R}^{m\times N}$, with the fundamental difference here being that the *mapped training data* $\phi(\mathbf{Y}) = [\phi(\mathbf{y}_1), \dots, \phi(\mathbf{y}_N)]$ are now assumed to be drawn from an MC-UoS $\mathcal{M}_L = \bigcup_{\ell=1}^{L} \mathcal{S}_\ell \subset \mathcal{G}_{\widetilde{m},s} \subset \mathcal{F}$. Here, we also make the simplified assumption that $\operatorname{rank}(\phi(\mathbf{Y})) = N$, which is justified as long as $\widetilde{m} \ggg N$ and no two training samples are identical. Our goal in this setting is to learn the (feature space) MC-UoS $\mathcal{M}_L$ using the training data $\mathbf{Y}$, which in theory can still be achieved by solving the following variant of (2.2):

$$\{\mathcal{S}_\ell\}_{\ell=1}^{L} = \underset{\{\mathcal{S}_\ell\}\subset\mathcal{G}_{\widetilde{m},s}}{\arg\min} \sum_{\substack{\ell,p=1\\ \ell\neq p}}^{L} d_u^2(\mathcal{S}_\ell, \mathcal{S}_p) + \lambda \sum_{i=1}^{N} \|\phi(\mathbf{y}_i) - P_{\mathcal{S}_{l_i}}\phi(\mathbf{y}_i)\|_2^2, \tag{3.1}$$

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

In practice, however, solving (3.1) 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" [15], which involves transforming (3.1) 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 κ, which is a positive semidefinite function $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ that satisfies $\kappa(\mathbf{y}, \mathbf{y}') = \langle \phi(\mathbf{y}), \phi(\mathbf{y}') \rangle$ for all $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^m$, to develop algorithms that can learn an MC-UoS in the feature space without explicit mapping of the training data to the feature space. We term the learning of an MC-UoS in the feature space using the kernel trick as metric-constrained kernel union-of-subspaces (MC-KUoS) learning. Similar to the case of MC-UoS learning, we consider two scenarios in this work for MC-KUoS learning. The first one of these scenarios corresponds to the standard setup in which all m dimensions of each training sample in $\mathbf{Y}$ are observed, while the second scenario corresponds to the case of "missing data" in which some dimensions of each training sample in $\mathbf{Y}$ remain unobserved. Finally, we will evaluate the proposed MC-KUoS learning algorithms using (i) the metric of average approximation error of noisy test data, and (ii) their clustering performance on training data having either complete or missing entries. We conclude here by pointing out that MC-KUoS learning invariably also leads us to the problem of finding the "pre-images" of data in the feature space induced by our chosen kernel (e.g., Gaussian or polynomial kernel) [6, 64], which will also be addressed in this chapter.

Remark 2. It is worth noting here that (3.1) requires knowledge of the nonlinear map ϕ. However, since we rely on the kernel trick for our MC-KUoS learning framework, we only need access to an appropriate kernel κ. It is assumed in this chapter that such a kernel is readily available to us. While learning the "best" kernel from training data is an interesting extension of our work, it is beyond the scope of this thesis.

In the following sections, we present algorithms to solve the problem of MC-KUoS learning from $\mathbf{Y} \in \mathbb{R}^{m \times N}$ for highly nonlinear data. We first generalize the MiCUSaL algorithm using the kernel trick [15] to learn an MC-KUoS from complete data. To deal with the case of "missing data," we propose "kernel function value estimators" to solve (3.1). Finally, we discuss the problem of finding the "pre-images" of data in the feature space based on the MC-KUoS model in Section 3.4.

3.2 MC-KUoS Learning Using Complete Data

To begin our discussion, we define the kernel matrix on the training data $\mathbf{Y}$ to be $\mathbf{G} = \phi(\mathbf{Y})^T\phi(\mathbf{Y}) \in \mathbb{R}^{N\times N}$, with its individual entries $g_{i,j} = \kappa(\mathbf{y}_i, \mathbf{y}_j)$ for a pre-specified kernel $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$. Under the assumption that $\operatorname{rank}(\phi(\mathbf{Y})) = N$, the matrix $\mathbf{G}$ is positive definite. Similar to Algorithm 1, we begin with centering the ϕ-mapped data in the feature space $\mathcal{F}$ as a pre-processing stage.[1] We denote the mean of the ϕ-mapped "images" of $\mathbf{Y}$ by $\overline{\boldsymbol{\phi}} = \frac{1}{N}\sum_{i=1}^N \phi(\mathbf{y}_i)$ and write the N centered "mapped training data" as $\widetilde{\phi}(\mathbf{Y}) = [\widetilde{\phi}(\mathbf{y}_1), \dots, \widetilde{\phi}(\mathbf{y}_N)]$, where $\widetilde{\phi}(\mathbf{y}_i) = \phi(\mathbf{y}_i) - \overline{\boldsymbol{\phi}}$, $i = 1, \dots, N$. The centered kernel matrix $\widetilde{\mathbf{G}} = \widetilde{\phi}(\mathbf{Y})^T\widetilde{\phi}(\mathbf{Y})$ can be calculated from $\mathbf{G}$ by [15]

$$\widetilde{\mathbf{G}} = \mathbf{G} - \mathbf{H}_N\mathbf{G} - \mathbf{G}\mathbf{H}_N + \mathbf{H}_N\mathbf{G}\mathbf{H}_N, \tag{3.2}$$

where $\mathbf{H}_N$ is an $N \times N$ matrix with all elements $\frac{1}{N}$. Then for any $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^m$, we have [64]

$$\widetilde{\kappa}(\mathbf{y}, \mathbf{y}') = \widetilde{\phi}(\mathbf{y})^T\widetilde{\phi}(\mathbf{y}') = \kappa(\mathbf{y}, \mathbf{y}') - \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}} - \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}'} + \frac{1}{N^2}\mathbf{1}_N^T\mathbf{G}\mathbf{1}_N, \tag{3.3}$$

where $\mathbf{1}_N = [1, 1, \dots, 1]^T$ is an N-dimensional vector and $\boldsymbol{k}_{\mathbf{y}} = [\kappa(\mathbf{y}, \mathbf{y}_1), \dots, \kappa(\mathbf{y}, \mathbf{y}_N)]^T \in \mathbb{R}^N$. To write the expression in (3.1) in terms of inner products, we again use $\mathbf{W}$ to denote the membership indicator matrix as in (2.3), where $w_{\ell,i} = 1$, $\ell = 1, \dots, L$, $i = 1, \dots, N$, if and only if $\widetilde{\phi}(\mathbf{y}_i)$ is assigned to subspace $\mathcal{S}_\ell$. Let $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_L]$, where $\mathbf{D}_\ell$ is an (arbitrary) orthonormal basis of $\mathcal{S}_\ell$. Then for any $i = 1, \dots, N$, we have the following

$$\|\phi(\mathbf{y}_i) - P_{\mathcal{S}_\ell}\phi(\mathbf{y}_i)\|_2^2 = \|\widetilde{\phi}(\mathbf{y}_i) - P_{\mathcal{S}_\ell}\widetilde{\phi}(\mathbf{y}_i)\|_2^2 = \|\widetilde{\phi}(\mathbf{y}_i)\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\phi}(\mathbf{y}_i)\|_2^2. \tag{3.4}$$

Therefore, (3.1) can be written as $(\mathbf{D}, \mathbf{W}) = \arg\min_{\mathbf{D},\mathbf{W}} F_3(\mathbf{D}, \mathbf{W})$ with the objective function $F_3(\mathbf{D}, \mathbf{W})$ given by

$$F_3(\mathbf{D}, \mathbf{W}) = \sum_{\substack{\ell,p=1\\ \ell\neq p}}^{L} \|\mathbf{D}_\ell - P_{\mathcal{S}_p}\mathbf{D}_\ell\|_F^2 + \lambda\sum_{i=1}^{N}\sum_{\ell=1}^{L} w_{\ell,i}(\|\widetilde{\phi}(\mathbf{y}_i)\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\phi}(\mathbf{y}_i)\|_2^2). \tag{3.5}$$

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

Before discussing our algorithm to minimize (3.5) using the kernel trick, we further simplify the terms in (3.5). We again define $\mathbf{c}_\ell = \{i \in \{1, \dots, N\} : w_{\ell,i} = 1\}$ to be the set containing the indices of all $\widetilde{\phi}(\mathbf{y}_i)$'s that are assigned to $\mathcal{S}_\ell$, and let $\mathbf{Y}_\ell = [\mathbf{y}_i : i \in \mathbf{c}_\ell]$ be the corresponding $m \times |\mathbf{c}_\ell|$ matrix. Then the centered data that are assigned to subspace $\mathcal{S}_\ell$ can be denoted by $\widetilde{\phi}(\mathbf{Y}_\ell) = [\widetilde{\phi}(\mathbf{y}_i) : i \in \mathbf{c}_\ell]$. Since $\mathcal{S}_\ell$ is spanned by the columns of $\widetilde{\phi}(\mathbf{Y}_\ell)$, we can write $\mathbf{D}_\ell = \widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell$, where $\mathbf{E}_\ell \in \mathbb{R}^{N_\ell \times s}$ is some basis representation matrix with $N_\ell = |\mathbf{c}_\ell|$ such that $\mathbf{D}_\ell$ is an orthonormal matrix. Also, it is easy to verify that $\mathbf{E}_\ell$ has to satisfy $\mathbf{E}_\ell^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{E}_\ell = \mathbf{I}_s$, where $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} = \widetilde{\phi}(\mathbf{Y}_\ell)^T\widetilde{\phi}(\mathbf{Y}_\ell)$ denotes the centered kernel matrix for subspace $\mathcal{S}_\ell$. Now instead of using $\mathbf{D}_\ell$ explicitly for computations, it suffices to use $\mathbf{c}_\ell$ and $\mathbf{E}_\ell$ for MC-KUoS learning and all the computations involving $\mathbf{D}_\ell$ can be carried out using $\mathbf{c}_\ell$, $\mathbf{E}_\ell$ and the kernel trick. Specifically, notice that for any $i = 1, \dots, N$, we can write

$$\|\widetilde{\phi}(\mathbf{y}_i)\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\phi}(\mathbf{y}_i)\|_2^2 = \widetilde{\kappa}(\mathbf{y}_i, \mathbf{y}_i) - \|\mathbf{E}_\ell^T\widetilde{\phi}(\mathbf{Y}_\ell)^T\widetilde{\phi}(\mathbf{y}_i)\|_2^2, \tag{3.6}$$

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

$$\begin{aligned} d_u^2(\mathcal{S}_\ell, \mathcal{S}_p) &= \|\mathbf{D}_\ell - P_{\mathcal{S}_p}\mathbf{D}_\ell\|_F^2 = s - \operatorname{tr}(\mathbf{D}_\ell^T\mathbf{D}_p\mathbf{D}_p^T\mathbf{D}_\ell) \\ &= s - \operatorname{tr}\Big[(\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell)^T\widetilde{\phi}(\mathbf{Y}_p)\mathbf{E}_p(\widetilde{\phi}(\mathbf{Y}_p)\mathbf{E}_p)^T\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell\Big] \\ &= s - \operatorname{tr}\Big[\mathbf{E}_\ell^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_p}\mathbf{E}_p\mathbf{E}_p^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_p,\mathbf{c}_\ell}\mathbf{E}_\ell\Big], \end{aligned} \tag{3.7}$$

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

Now we are ready to describe our algorithm in detail. Similar to MiCUSaL, we minimize (3.5) by alternating between (i) minimizing $F_3(\mathbf{D}, \mathbf{W})$ over $\mathbf{W}$ for a fixed $\mathbf{D}$ (the *kernel subspace assignment* step) and (ii) minimizing $F_3(\mathbf{D}, \mathbf{W})$ over $\mathbf{D}$ for a fixed $\mathbf{W}$ (the *kernel subspace update* step). To begin this alternate optimization strategy, we start by initializing the orthonormal basis of each subspace. As discussed

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

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

earlier, the orthonormal basis $\mathbf{D}_\ell$ of $\mathcal{S}_\ell$ can be represented as $\mathbf{D}_\ell = \widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell$ and we can compute $\mathbf{E}_\ell$ explicitly by using $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell, \mathbf{c}_\ell}$. Therefore the initialization of $\mathbf{D}_\ell$ is equivalent to initializing $\mathbf{c}_\ell$. Note that any s linear independent vectors define an s-dimensional subspace. Therefore, to initialize $\mathbf{c}_\ell$, we need to choose s samples in the training set such that the ϕ-mapped "images" of these training samples are linearly independent in the feature space. Our selection of s samples is based on the intuition that the inner products between samples that lie in the same subspace in the feature space will be typically large [33]. Our initialization procedure then involves greedily selecting a new sample $\mathbf{y}_{i^*}$ from the training data in each step such that the sum of the inner products between $\widetilde{\phi}(\mathbf{y}_{i^*})$ and the data points already in $\widetilde{\phi}(\mathbf{Y}_\ell)$ is the largest, and then setting $\mathbf{Y}_\ell = \mathbf{Y}_\ell \cup \mathbf{y}_{i^*}$. We list our initialization method in Algorithm 4, referred to as *greedy kernel initial-orthogonalization procedure* (GKIOP). Based on the assumption that all the $\phi(\mathbf{y}_i)$'s are linearly independent, it is guaranteed that $\widetilde{\phi}(\mathbf{Y}_\ell)$ can define an s-dimensional subspace $\mathcal{S}_\ell$. Note that $\bigcap_{\ell=1}^{L} \mathbf{c}_\ell = \emptyset$ and we compute $\mathbf{E}_\ell$ by $\mathbf{E}_\ell = \mathbf{U}_\ell \mathbf{\Sigma}_\ell^{-\frac{1}{2}}$, where $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell, \mathbf{c}_\ell} = \mathbf{U}_\ell \mathbf{\Sigma}_\ell \mathbf{U}_\ell^T$. Since $\mathbf{D}_\ell = \widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell$, it is easy to convince oneself that $\mathbf{D}_\ell^T \mathbf{D}_\ell = \mathbf{I}_s$ in this case.

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

$$l_i = \operatorname*{arg\,min}_{\ell=1,\dots,L} \|\widetilde{\phi}(\mathbf{y}_i) - P_{\mathcal{S}_\ell}\widetilde{\phi}(\mathbf{y}_i)\|_2^2 = \operatorname*{arg\,min}_{\ell=1,\dots,L} \widetilde{\kappa}(\mathbf{y}_i, \mathbf{y}_i) - \|\mathbf{E}_\ell^T \widetilde{\boldsymbol{\psi}}_\ell(\mathbf{y}_i)\|_2^2, \tag{3.8}$$

and then setting $w_{l_i,i} = 1$ and $w_{\ell,i} = 0$ $\forall \ell \neq l_i$. Next, for the kernel subspace update stage, since $\mathbf{W}$ is fixed, all the $\mathbf{c}_\ell$'s and $\mathbf{Y}_\ell$'s are fixed. By writing $\mathbf{D}_\ell = \widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell$, minimization of (3.5) for a fixed $\mathbf{W}$ can be written as a function of $\mathbf{E}_\ell$'s as follows:

$$\begin{aligned}\min_{\{\mathbf{E}_\ell\}} f_3(\mathbf{E}_1, \dots, \mathbf{E}_L) = & \sum_{\substack{\ell,p=1\\ \ell\neq p}}^{L} \|\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell - P_{\mathcal{S}_p}(\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell)\|_F^2 \\ & + \lambda \sum_{\ell=1}^{L} \big(\|\widetilde{\phi}(\mathbf{Y}_\ell)\|_F^2 - \|\mathbf{E}_\ell^T \widetilde{\phi}(\mathbf{Y}_\ell)^T \widetilde{\phi}(\mathbf{Y}_\ell)\|_F^2\big) \\ \text{s.t.} \quad & \mathbf{E}_\ell^T [\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} \mathbf{E}_\ell = \mathbf{I}_s, \;\; \ell = 1, 2, \dots, L. \end{aligned} \tag{3.9}$$

Instead of updating all the $\mathbf{E}_\ell$'s simultaneously, which is again a difficult optimization problem, we use BCD to minimize f_3 and update $\mathbf{E}_\ell$'s sequentially. Unlike MC-UoS learning, however, we have to be careful here since the number of samples in $\mathbf{Y}$ that belong to $\mathbf{Y}_\ell$ (i.e., N_ℓ) may change after each subspace assignment step. In particular, we first need to initialize all the $\mathbf{E}_\ell$'s such that $\mathbf{E}_\ell \in \mathbb{R}^{N_\ell \times s}$ and $\mathbf{E}_\ell^T [\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} \mathbf{E}_\ell = \mathbf{I}_s$. To do so, we again apply eigendecomposition of $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} = \mathbf{U}_\ell \boldsymbol{\Sigma}_\ell \mathbf{U}_\ell^T$ with the diagonal entries of $\boldsymbol{\Sigma}_\ell$ in nonincreasing order. Then we define $\boldsymbol{\mathcal{I}}_s = \{1, \dots, s\}$ and $\mathbf{E}_\ell = [\mathbf{U}_\ell]_{:,\boldsymbol{\mathcal{I}}_s} [\boldsymbol{\Sigma}_\ell]_{\boldsymbol{\mathcal{I}}_s,\boldsymbol{\mathcal{I}}_s}^{-\frac{1}{2}}$. After this bases initialization step, we are ready to update $\mathbf{E}_\ell$'s sequentially and after some manipulations, each BCD subproblem of (3.9) can be expressed as

$$\begin{aligned}\mathbf{E}_\ell = & \operatorname*{arg\,min}_{\mathbf{Q}:\mathbf{Q}^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{Q}=\mathbf{I}_s} \sum_{p\neq\ell} \|\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{Q} - P_{\mathcal{S}_p}(\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{Q})\|_F^2 \\ & \qquad + \frac{\lambda}{2}(\|\widetilde{\phi}(\mathbf{Y}_\ell)\|_F^2 - \|\mathbf{Q}^T \widetilde{\phi}(\mathbf{Y}_\ell)^T \widetilde{\phi}(\mathbf{Y}_\ell)\|_F^2) \\ = & \operatorname*{arg\,max}_{\mathbf{Q}:\mathbf{Q}^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{Q}=\mathbf{I}_s} \operatorname{tr}(\mathbf{Q}^T \mathbf{A}_\ell \mathbf{Q}), \end{aligned} \tag{3.10}$$

where $\mathbf{A}_\ell = \sum_{p\neq\ell} [\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_p} \mathbf{E}_p \mathbf{E}_p^T [\widetilde{\mathbf{G}}]_{\mathbf{c}_p,\mathbf{c}_\ell} + \frac{\lambda}{2}[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}^2$ is a symmetric matrix of dimension $N_\ell \times N_\ell$. Note that (3.10) has a similar intuitive interpretation as (2.7). When $\lambda = 0$, (3.10) reduces to the problem of finding a subspace that is closest to the remaining $L-1$ subspaces in the feature space. When $\lambda = \infty$, (3.10) reduces to the kernel PCA

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

```
Input: Training data $\mathbf{Y} \in \mathbb{R}^{m\times N}$, number and dimension of subspaces $L$ and $s$,
  kernel function $\kappa$ and parameter $\lambda$.
 1: Compute kernel matrix $\mathbf{G}$: $g_{i,j} = \kappa(\mathbf{y}_i, \mathbf{y}_j)$.
 2: $\widetilde{\mathbf{G}} = \mathbf{G} - \mathbf{H}_N\mathbf{G} - \mathbf{G}\mathbf{H}_N + \mathbf{H}_N\mathbf{G}\mathbf{H}_N$.
 3: Initialize $\{\mathbf{c}_\ell\}_{\ell=1}^L$ and $\{\mathbf{E}_\ell\}_{\ell=1}^L$ using GKIOP (Algorithm 4).
 4: while stopping rule do
 5:   for $i = 1$ to $N$ (Kernel Subspace Assignment) do
 6:     $l_i = \arg\min_\ell \widetilde{\kappa}(\mathbf{y}_i, \mathbf{y}_i) - \|\mathbf{E}_\ell^T \widetilde{\boldsymbol{\psi}}_\ell(\mathbf{y}_i)\|_2^2$.
 7:     $w_{l_i,i} = 1$ and $\forall \ell \neq l_i$, $w_{\ell,i} = 0$.
 8:   end for
 9:   for $\ell = 1$ to $L$ (Kernel Bases Initialization) do
10:     $\mathbf{c}_\ell = \{i \in \{1,\dots,N\} : w_{\ell,i} = 1\}$ and $N_\ell = |\mathbf{c}_\ell|$.
11:     Eigendecomposition of $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} = \mathbf{U}_\ell \boldsymbol{\Sigma}_\ell \mathbf{U}_\ell^T$, with the diagonal elements of $\boldsymbol{\Sigma}_\ell$ in
        nonincreasing order.
12:     $\mathbf{E}_\ell = [\mathbf{U}_\ell]_{:,\mathcal{I}_s} [\boldsymbol{\Sigma}_\ell]_{\mathcal{I}_s,\mathcal{I}_s}^{-\frac{1}{2}}$.
13:   end for
14:   while stopping rule do
15:     for $\ell = 1$ to $L$ (Kernel Subspace Update) do
16:       $\mathbf{A}_\ell = \sum_{p\neq\ell} [\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_p} \mathbf{E}_p \mathbf{E}_p^T [\widetilde{\mathbf{G}}]_{\mathbf{c}_p,\mathbf{c}_\ell} + \frac{\lambda}{2} [\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}^2$.
17:       $\mathbf{E}_\ell$ = Eigenvectors corresponding to the $s$-largest eigenvalues for the
          generalized problem $\mathbf{A}_\ell \mathbf{b} = \zeta [\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} \mathbf{b}$ such that $\mathbf{E}_\ell^T [\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} \mathbf{E}_\ell = \mathbf{I}_s$.
18:     end for
19:   end while
20: end while
Output: $\{N_\ell \in \mathbb{N}\}_{\ell=1}^L$, $\{\mathbf{c}_\ell\}_{\ell=1}^L$ and $\{\mathbf{E}_\ell \in \mathbb{R}^{N_\ell \times s}\}_{\ell=1}^L$.
```

problem [26]. Since $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}$ is a positive definite matrix, it again follows from [54] that the trace of $\mathbf{E}_\ell^T \mathbf{A}_\ell \mathbf{E}_\ell$ is maximized when $\mathbf{E}_\ell = [\mathbf{b}_1, \dots, \mathbf{b}_s]$ is the set of eigenvectors associated with the s-largest eigenvalues for the generalized problem $\mathbf{A}_\ell \mathbf{b} = \zeta [\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} \mathbf{b}$, with $\mathbf{E}_\ell^T [\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} \mathbf{E}_\ell = \mathbf{I}_s$. The whole algorithm can be detailed in Algorithm 5, which we refer to as *metric-constrained kernel union-of-subspaces learning* (MC-KUSaL). An important thing to notice here is that the complexity of MC-KUSaL does not scale with the dimensionality of the feature space $\mathcal{F}$ owing to our use of the kernel trick.

3.3 MC-KUoS Learning Using Missing Data

In this section, we focus on MC-KUoS learning for the case of training data having missing entries in the input space. Our setup is similar to the one in Section 2.4.

That is, for $i = 1, \dots, N$, we observe $\mathbf{y}_i$ only at locations $\mathbf{\Omega}_i \subset \{1, \dots, m\}$. In the following, the resulting observed vector of $\mathbf{y}_i$ is denoted by $[\mathbf{y}_i]_{\mathbf{\Omega}_i} \in \mathbb{R}^{|\mathbf{\Omega}_i|}$. Also, we assume that the observed indices of each signal, $\mathbf{\Omega}_i$, are drawn uniformly at random with replacement from $\{1, \dots, m\}$. Note that the results derived in here can also be translated to the case of sampling $\mathbf{\Omega}_i$ without replacement (we refer the reader to [65, Lemma 1] as an example). Given the missing data aspect of this setup and the kernel trick, it is clear that we cannot apply the method in Section 2.4 for MC-KUoS learning. However, as described in Section 3.2, the solution to the MC-KUoS learning problem using complete data only requires computations of the inner products in $\mathcal{F}$. In this regard, we propose an estimate of the kernel function value $\kappa(\mathbf{y}_i, \mathbf{y}_j)$ using incomplete data $[\mathbf{y}_i]_{\mathbf{\Omega}_i}$ and $[\mathbf{y}_j]_{\mathbf{\Omega}_j}$. Mathematically, our goal is to find a proxy function $h(\cdot, \cdot)$ such that $h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j}) \approx \kappa(\mathbf{y}_i, \mathbf{y}_j)$. To derive this proxy function, we start by considering the relationship between $[\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j}$ and $\mathbf{y}_i, \mathbf{y}_j$ in the context of different kernel functions.

We first consider isotropic kernels of the form $\kappa(\mathbf{y}_i, \mathbf{y}_j) = k(\|\mathbf{y}_i - \mathbf{y}_j\|_2^2)$ for our analysis. To begin, we define $\mathbf{z}_{ij}^- = \mathbf{y}_i - \mathbf{y}_j$ and $\mathbf{\Omega}_{ij} = \mathbf{\Omega}_i \cap \mathbf{\Omega}_j$, resulting in $[\mathbf{z}_{ij}^-]_{\mathbf{\Omega}_{ij}} = [\mathbf{y}_i]_{\mathbf{\Omega}_{ij}} - [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}} \in \mathbb{R}^{|\mathbf{\Omega}_{ij}|}$. For any vector $\mathbf{z}_{ij}^-$, the authors in [59] define the coherence of a subspace spanned by a vector $\mathbf{z}_{ij}^-$ to be $\mu(\mathbf{z}_{ij}^-) = \frac{m\|\mathbf{z}_{ij}^-\|_\infty^2}{\|\mathbf{z}_{ij}^-\|_2^2}$ and show that $\|[\mathbf{z}_{ij}^-]_{\mathbf{\Omega}_{ij}}\|_2^2$ is close to $\frac{|\mathbf{\Omega}_{ij}|}{m}\|\mathbf{z}_{ij}^-\|_2^2$ with high probability. Leveraging this result, we can give the following corollary that is essentially due to [59, Lemma 1] by plugging in the definition of $\mathbf{z}_{ij}^-$ and $[\mathbf{z}_{ij}^-]_{\mathbf{\Omega}_{ij}}$.

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

$$(1-\alpha)\|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \le \frac{m}{|\mathbf{\Omega}_{ij}|}\|[\mathbf{z}_{ij}^-]_{\mathbf{\Omega}_{ij}}\|_2^2 \le (1+\alpha)\|\mathbf{y}_i - \mathbf{y}_j\|_2^2. \tag{3.11}$$

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

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

$$h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j})^{\frac{1}{1-\alpha}} \leq \kappa(\mathbf{y}_i, \mathbf{y}_j) \leq h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j})^{\frac{1}{1+\alpha}}. \tag{3.12}$$

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

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

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

$$\langle \mathbf{y}_i, \mathbf{y}_j \rangle - \beta \leq \frac{m}{|\mathbf{\Omega}_{ij}|}\langle [\mathbf{y}_i]_{\mathbf{\Omega}_{ij}}, [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}} \rangle \leq \langle \mathbf{y}_i, \mathbf{y}_j \rangle + \beta. \tag{3.13}$$

Proof. First, we have $\langle \mathbf{y}_i, \mathbf{y}_j \rangle = \sum_{u=1}^m z^*_{ij(u)}$ and $\langle [\mathbf{y}_i]_{\mathbf{\Omega}_{ij}}, [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}} \rangle = \sum_{v=1}^n z^*_{ij(\Omega_{ij(v)})}$ with $n = |\mathbf{\Omega}_{ij}|$. Here, $z^*_{ij(u)}$ denotes the u-th entry of a vector $\mathbf{z}^*_{ij}$ and $\Omega_{ij(v)}$ denotes the v-th element of $\mathbf{\Omega}_{ij}$. Let $\hbar(Z_1, \ldots, Z_n) = \sum_{v=1}^n Z_v$ be the sum of n random variables and $Z_v = z^*_{ij(\Omega_{ij(v)})}$. 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 $\mathbb{E}[\sum_{v=1}^n Z_v] = \mathbb{E}[\sum_{v=1}^n z^*_{ij(\Omega_{ij(v)})}] = \frac{n}{m}\sum_{u=1}^m z^*_{ij(u)}$. If the value of one variable in the sum is replaced by any other of its possible values, the sum changes at most $2\|\mathbf{z}^*_{ij}\|_\infty$, i.e., $|\sum_{v=1}^n Z_v - \sum_{v \neq v'} Z_v - \widehat{Z}_{v'}| = |Z_{v'} - \widehat{Z}_{v'}| \leq 2\|\mathbf{z}^*_{ij}\|_\infty$ for any $v' \in \{1, \ldots, n\}$. Therefore, McDiarmid's Inequality [66] implies that for $\beta > 0$,

$$\mathbb{P}\Big[|\sum_{v=1}^n Z_v - \frac{n}{m}\sum_{u=1}^m z^*_{ij(u)}| \geq \frac{n}{m}\beta\Big] \leq 2\exp\Big(\frac{-n\beta^2}{2m^2\|\mathbf{z}^*_{ij}\|^2_\infty}\Big), \tag{3.14}$$

or equivalently,

$$\mathbb{P}\Big[\sum_{u=1}^{m} z^*_{ij(u)} - \beta \leq \frac{m}{n}\sum_{v=1}^{n} Z_v \leq \sum_{u=1}^{m} z^*_{ij(u)} + \beta\Big] \geq 1 - 2\exp\Big(\frac{-n\beta^2}{2m^2\|\mathbf{z}^*_{ij}\|^2_\infty}\Big). \tag{3.15}$$

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

The above lemma establishes that $\langle[\mathbf{y}_i]_{\mathbf{\Omega}_{ij}}, [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}}\rangle$ is close to $\frac{|\mathbf{\Omega}_{ij}|}{m}\langle\mathbf{y}_i, \mathbf{y}_j\rangle$ with high probability. We once again use this relationship and give an estimate of the corresponding kernel function value. For example, for the polynomial kernel $\kappa(\mathbf{y}_i, \mathbf{y}_j) = (\langle\mathbf{y}_i, \mathbf{y}_j\rangle + c)^d$ with $d > 0$ and $c \geq 0$, we have $h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j}) = (\frac{m}{|\mathbf{\Omega}_{ij}|}\langle[\mathbf{y}_i]_{\mathbf{\Omega}_{ij}}, [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}}\rangle + c)^d$. To analyze the bounds on estimated kernel function value in this case, notice that if (3.13) holds and d is odd, we will have

$$(\langle\mathbf{y}_i, \mathbf{y}_j\rangle - \beta + c)^d \leq (\frac{m}{|\mathbf{\Omega}_{ij}|}\langle[\mathbf{y}_i]_{\mathbf{\Omega}_{ij}}, [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}}\rangle + c)^d \leq (\langle\mathbf{y}_i, \mathbf{y}_j\rangle + \beta + c)^d. \tag{3.16}$$

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

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

$$(h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j})^{\frac{1}{d}} - \beta)^d \leq \kappa(\mathbf{y}_i, \mathbf{y}_j) \leq (h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j})^{\frac{1}{d}} + \beta)^d. \tag{3.17}$$

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

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

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

3.4 Pre-Image Reconstruction

Thus far in this chapter, we have discussed MC-UoS learning in the kernel space with complete and missing data using the kernel trick. Now suppose we are given a new noisy (test) sample $\mathbf{z} = \mathbf{x} + \boldsymbol{\xi} \in \mathbb{R}^m$, where $\boldsymbol{\xi}$ is a noise term and $\widetilde{\phi}(\mathbf{x}) = \phi(\mathbf{x}) - \overline{\boldsymbol{\phi}}$ belongs to one of the subspaces in $\mathcal{M}_L$ (i.e., $\widetilde{\phi}(\mathbf{x}) \in \mathcal{S}_\tau, \tau \in \{1, \ldots, L\}$). In most information processing tasks, one needs to first find a representation of this sample $\mathbf{z}$ in terms of the learned MC-KUoS, which is akin to "denoising" $\mathbf{z}$. The "denoised sample" in the feature space is the projection of $\phi(\mathbf{z})$ onto $\mathcal{S}_\tau$, which is given by $P_{\mathcal{S}_\tau}\phi(\mathbf{z}) = \mathbf{D}_\tau \mathbf{D}_\tau^T \widetilde{\phi}(\mathbf{z}) + \overline{\boldsymbol{\phi}}$ with $\widetilde{\phi}(\mathbf{z}) = \phi(\mathbf{z}) - \overline{\boldsymbol{\phi}}$. However, in order to visualize the "denoised" sample in the ambient space, we often need to project $P_{\mathcal{S}_\tau}\phi(\mathbf{z})$ onto the input space in many applications [6,67], which is termed *pre-image reconstruction*. In this section, we consider the problem of pre-image reconstruction based on the MC-KUoS model.

Mathematically, the problem of pre-image reconstruction can be stated as follows. We are given $\mathbf{z} \in \mathbb{R}^m$ and we are interested in finding $\widehat{\mathbf{z}} \in \mathbb{R}^m$ whose mapping to the feature space is closest to the projection of $\phi(\mathbf{z})$ onto the learned MC-UoS in $\mathcal{F}$. This involves first finding the index τ such that $\tau = \arg\min_\ell \|\widetilde{\phi}(\mathbf{z}) - P_{\mathcal{S}_\ell}\widetilde{\phi}(\mathbf{z})\|_2^2$, which can be easily done using the kernel subspace assignment step described in (3.8). Next, we need to solve $\widehat{\mathbf{z}} = \arg\min_{\boldsymbol{\varrho}\in\mathbb{R}^m} \|\phi(\boldsymbol{\varrho}) - P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2$. To solve this problem, we leverage the ideas in [64,68] that only use feature-space distances to find $\widehat{\mathbf{z}}$ (equivalently, to find the pre-image of $P_{\mathcal{S}_\tau}\phi(\mathbf{z})$). We first study this problem when the training samples $\mathbf{Y}$ are complete.

3.4.1 Pre-Image Reconstruction Using Complete Data

We first calculate the squared "feature distance" between $P_{\mathcal{S}_\tau}\phi(\mathbf{z})$ and any $\phi(\mathbf{y}_i)$, $i = 1\dots,N$, defined as [64]

$$d_{\mathcal{F}}^2(\phi(\mathbf{y}_i), P_{\mathcal{S}_\tau}\phi(\mathbf{z})) = \|P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2 + \|\phi(\mathbf{y}_i)\|_2^2 - 2(P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\mathbf{y}_i). \tag{3.18}$$

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

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

and

$$(P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\mathbf{y}_i) = \widetilde{\boldsymbol{\psi}}_\tau(\mathbf{z})^T\mathbf{E}_\tau\mathbf{E}_\tau^T\Big(\boldsymbol{\psi}_\tau(\mathbf{y}_i) - \frac{1}{N}\mathbf{1}_{N_\tau}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}_i}\Big) + \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}_i}. \tag{3.19}$$

Therefore, (3.18) becomes

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

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

We now describe our method for pre-image reconstruction using the Gaussian kernel $\kappa(\mathbf{y}_i, \mathbf{y}_j) = \exp(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|_2^2}{c})$ first. In this case, the problem of minimizing $\|\phi(\widehat{\mathbf{z}}) - P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2$ is equivalent to maximizing the function $\rho(\widehat{\mathbf{z}}) = (P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\widehat{\mathbf{z}})$ [6], whose extremum can be obtained by setting $\nabla_{\widehat{\mathbf{z}}}\rho = 0$, where $\nabla_{\widehat{\mathbf{z}}}\rho$ denotes the gradient of ρ with respect to $\widehat{\mathbf{z}}$. To do so, we express $\rho(\widehat{\mathbf{z}})$ as

$$\begin{aligned}
\rho(\widehat{\mathbf{z}}) &= (\mathbf{D}_\tau\mathbf{D}_\tau^T\widetilde{\phi}(\mathbf{z}) + \overline{\boldsymbol{\phi}})^T\phi(\widehat{\mathbf{z}}) \\
&= \widetilde{\phi}(\mathbf{z})^T\widetilde{\phi}(\mathbf{Y}_\tau)\mathbf{E}_\tau\mathbf{E}_\tau^T\widetilde{\phi}(\mathbf{Y}_\tau)^T\phi(\widehat{\mathbf{z}}) + \frac{1}{N}\mathbf{1}_N^T\phi(\mathbf{Y})^T\phi(\widehat{\mathbf{z}}) \\
&= \widetilde{\boldsymbol{\psi}}_\tau(\mathbf{z})^T\mathbf{E}_\tau\mathbf{E}_\tau^T(\boldsymbol{\psi}_\tau(\widehat{\mathbf{z}}) - \frac{1}{N}\mathbf{1}_{N_\tau}\mathbf{1}_N^T\boldsymbol{k}_{\widehat{\mathbf{z}}}) + \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\widehat{\mathbf{z}}} \\
&= \boldsymbol{\zeta}_\tau(\mathbf{z})^T(\boldsymbol{\psi}_\tau(\widehat{\mathbf{z}}) - \frac{1}{N}\mathbf{1}_{N_\tau}\mathbf{1}_N^T\boldsymbol{k}_{\widehat{\mathbf{z}}}) + \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\widehat{\mathbf{z}}},
\end{aligned} \tag{3.20}$$

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

$$\widehat{\mathbf{z}} = \frac{\sum_{i=1}^N \widehat{\chi}_{(i)}\exp(-\|\widehat{\mathbf{z}} - \mathbf{y}_i\|_2^2/c)\mathbf{y}_i}{\sum_{i=1}^N \widehat{\chi}_{(i)}\exp(-\|\widehat{\mathbf{z}} - \mathbf{y}_i\|_2^2/c)}. \tag{3.21}$$

By using the approximation $P_{\mathcal{S}_\tau}\phi(\mathbf{z}) \approx \phi(\widehat{\mathbf{z}})$ and the relation $\|\widehat{\mathbf{z}} - \mathbf{y}_i\|_2^2 = -c\log(\frac{1}{2}(2 - d_{\mathcal{F}}^2(\phi(\mathbf{y}_i), \phi(\widehat{\mathbf{z}}))))$ [64], a unique pre-image can now be obtained by the following formula:

$$\widehat{\mathbf{z}} = \frac{\sum_{i=1}^N \widehat{\chi}_{(i)}\Big(\frac{1}{2}\big(2 - d_{\mathcal{F}}^2(\phi(\mathbf{y}_i), P_{\mathcal{S}_\tau}\phi(\mathbf{z}))\big)\Big)\mathbf{y}_i}{\sum_{i=1}^N \widehat{\chi}_{(i)}\Big(\frac{1}{2}\big(2 - d_{\mathcal{F}}^2(\phi(\mathbf{y}_i), P_{\mathcal{S}_\tau}\phi(\mathbf{z}))\big)\Big)}. \tag{3.22}$$

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

$$\widehat{\mathbf{z}} = \sum_{i=1}^N \widehat{\chi}_{(i)}\Big(\frac{(P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\mathbf{y}_i)}{\|P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2}\Big)^{\frac{d-1}{d}}\mathbf{y}_i. \tag{3.23}$$

3.4.2 Pre-Image Reconstruction Using Missing Data

We next consider the problem of reconstructing the pre-image of $P_{\mathcal{S}_\tau}\phi(\mathbf{z})$ when the training samples have missing entries. As can be easily seen from (3.22), the solution of a pre-image for the Gaussian kernel can be written as $\widehat{\mathbf{z}} = \frac{\sum_{i=1}^{N} e_i \mathbf{y}_i}{\sum_{i=1}^{N} e_i}$, where $e_i = \widehat{\chi}_{(i)}\big(\frac{1}{2}(2 - d_{\mathcal{F}}^2(P_{\mathcal{S}_\tau}\phi(\mathbf{z}), \phi(\mathbf{y}_i)))\big)$. Similarly, from (3.23), we can also write the solution of $\widehat{\mathbf{z}}$ to be $\widehat{\mathbf{z}} = \sum_{i=1}^{N} e_i \mathbf{y}_i$ for the polynomial kernel, where $e_i = \widehat{\chi}_{(i)}\big(\frac{(P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T \phi(\mathbf{y}_i)}{\|P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2}\big)^{\frac{d-1}{d}}$ in this case. In words, the pre-image solution is a linear combination of the training data, where the weights e_i's can be explicitly computed using the respective kernel functions. In this regard, as described in Section 3.3, for each $i = 1, \dots, N$, we can estimate $\kappa(\mathbf{z}, \mathbf{y}_i)$ using entries of $\mathbf{z}$ belonging to $\mathbf{\Omega}_i$ (i.e., $[\mathbf{z}]_{\mathbf{\Omega}_i}$) and $[\mathbf{y}_i]_{\mathbf{\Omega}_i}$, where the estimated kernel function value is denoted by $h(\mathbf{z}, [\mathbf{y}_i]_{\mathbf{\Omega}_i})$.

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

3.5 Experimental Results

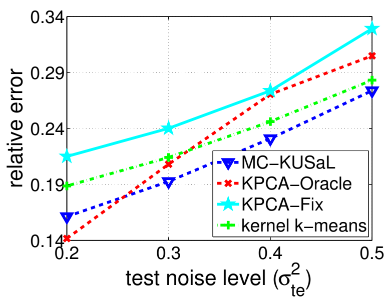
In this section, we evaluate the performance of the MC-KUoS learning approaches in terms of the following two problems: image denoising using the learned MC-KUoS and clustering of training data points. For both these problems, we consider the USPS dataset [69], which contains a collection of $m = 256$-dimensional handwritten digits. The authors in [6] have demonstrated that using nonlinear features can improve the denoising performance of this dataset. Unlike the experiments for MC-UoS learning, the training data we use in this set of experiments are noiseless. For denoising experiments, we assume every noisy *test* sample $\mathbf{z} = \mathbf{x} + \boldsymbol{\xi}$, where $\widetilde{\phi}(\mathbf{x}) = \phi(\mathbf{x}) - \overline{\boldsymbol{\phi}}$ belongs to one of the $\mathcal{S}_\ell$'s in $\mathcal{F}$ (again $\|\mathbf{x}\|_2^2 = 1$) and $\boldsymbol{\xi}$ has $\mathcal{N}(\mathbf{0}, (\sigma_{te}^2/m)\mathbf{I}_m)$ distribution. In these experiments, σ_{te}^2 ranges from 0.2 to 0.5.

3.5.1 Experiments on Image Denoising

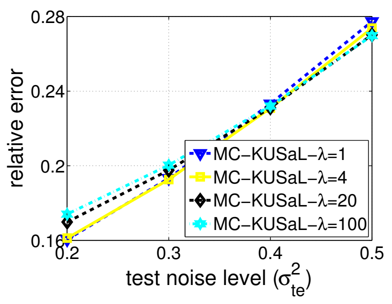
For denoising experiments, we compare the result of MC-KUSaL with three other methods: (*i*) kernel k-means clustering (kernel k-means) [15], where for each test signal $\mathbf{z}$, we first assign $\phi(\mathbf{z})$ to a cluster whose centroid is closest to $\phi(\mathbf{z})$ in $\mathcal{F}$, followed by kernel PCA and the method in [68] to calculate the pre-image; (*ii*) kernel PCA [26] with the same number of eigenvectors as in MC-KUSaL (KPCA-Fix); and (*iii*) kernel PCA with the number of eigenvectors chosen by $s = \arg\min_s ||P_{\mathcal{S}}\phi(\mathbf{z}) - \phi(\mathbf{x})||_2^2$ (KPCA-Oracle), where $\mathbf{x}$ and $\mathbf{z}$ are clean and noisy test samples, respectively. In this manner, the number of eigenvectors s for KPCA-Oracle will be different for different noise levels σ_{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 σ_{te}, we set the dimension of the subspace s for rKPCA to be the same as the one for KPCA-Oracle. The relative reconstruction error of a clean test signal $\mathbf{x} \in \mathbf{X}^{te}$ is calculated by $\frac{\|\mathbf{x}-\widehat{\mathbf{z}}\|_2^2}{\|\mathbf{x}\|_2^2}$, where $\widehat{\mathbf{z}}$ denotes the pre-image with respect to the noisy test sample $\mathbf{z}$.

We experiment with Gaussian kernel with parameter $c = 4$. We choose the digits "0" and "4" and for each digit we select the first 200 samples in the dataset (400 images in total) for our experiments. All these 400 samples are then vectorized and normalized to unit ℓ_2 norms. From these samples, we randomly choose 120 samples (without replacement) from each class for training and the remaining 80 samples of each class for testing, forming $\mathbf{X} \in \mathbb{R}^{256\times240}$ and $\mathbf{X}^{te} \in \mathbb{R}^{256\times160}$. This random selection of test and training samples is repeated 20 times for cross-validation purposes. We perform 10 Monte Carlo trials for noisy test data and report the mean over these 200 random trials.

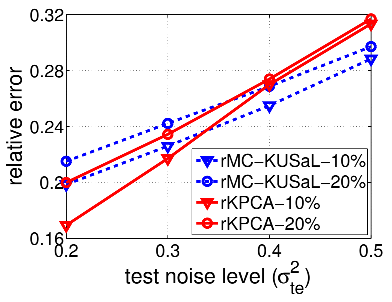
In these experiments, we implement MC-KUSaL with parameters $L = 2$, $s = 45$ and $\lambda \in \{1, 4, 20, 100\}$ to learn an MC-UoS in the feature space $\mathcal{F}$. Fig. 3.1(a) shows the mean of relative reconstruction errors of test data for different methods in the presence of complete training data, where we use the result of MC-KUSaL with $\lambda = 4$ for



(a) Complete data



(b) Complete data



(c) Missing data

Figure 3.1: Comparison of MC-KUoS learning performance on USPS dataset using Gaussian kernel $\kappa(\mathbf{y}, \mathbf{y}') = \exp(-\frac{\|\mathbf{y}-\mathbf{y}'\|_2^2}{4})$. In (a), we perform MC-KUSaL with $\lambda = 4$. Note that the KPCA-Oracle algorithm is the ideal case of kernel PCA. The numbers in the legend of (c) indicate the percentages of missing entries within the training data.

comparison with other methods. We observe that for almost all noise levels, our method produces better results than other methods. The only exception is when $\sigma_{te}^2 = 0.2$, in which case MC-KUSaL is the second best of all methods. The caveat here is that in practice, we cannot know beforehand the dimension of the subspace in the feature space for kernel PCA, which yields the best denoising result at this particular noise level. We show the denoising performance of MC-KUSaL with different λ's in Fig. 3.1(b), and we observe that a small λ usually results in good performance when σ_{te}^2 is relatively small, while increasing the λ will slightly improve the denoising performance when the SNR of test data gets small.

In the missing data experiments, we set the number of missing entries in the training data to be 10% and 20% of the signal dimension. We use parameters $L = 2$, $s = 45$ and $\lambda = 4$ for rMC-KUSaL. It can be inferred from Fig. 3.1(c) that (i) the performance of rKPCA and rMC-KUSaL is comparable for all noise levels; and (ii) when the number

Table 3.1: Clustering error (%) of the USPS dataset

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

of missing elements is fixed, rMC-KUSaL outperforms the rKPCA when the SNR of the test data is small and vice versa.

3.5.2 Experiments on Clustering

In this section, we empirically compare the clustering performance of MC-KUSaL with (*i*) kernel k-means clustering (kernel k-means) [15], (*ii*) standard k-means clustering (k-means) [70], and (*iii*) spectral clustering [71] when the training data are complete. In the case of spectral clustering, we make use of the similarity matrix returned by the noisy variant of the SSC optimization program in [13]. We also present the clustering performance of rMC-KUSaL, with the number of missing entries in the training data being set to 10% and 20% of the signal dimension. We compute the clustering error for MC-KUSaL/rMC-KUSaL by using the final kernel subspace assignment labels $\{l_i\}_{i=1}^N$. For all the following experiments, we select $L = 2$ (since we only have 2 classes) and $\lambda = 200$.

We first experiment with digits "1" and "7" in the USPS dataset, where in every trial we randomly choose 120 ℓ_2 normalized samples from the first 200 samples of these two digits and use these 240 samples in these experiments. This random selection is repeated 20 times. We perform MC-KUSaL and rMC-KUSaL using Gaussian kernel $\kappa(\mathbf{y},\mathbf{y}') = \exp(-\frac{\|\mathbf{y}-\mathbf{y}'\|_2^2}{8})$ with $s = 35$ and polynomial kernel $\kappa(\mathbf{y},\mathbf{y}') = (\langle \mathbf{y},\mathbf{y}' \rangle + 2)^3$ with $s = 40$. The parameter α_z for SSC is set to be 20. The clustering results are listed in Table 3.1, where we can see the clustering error for MC-KUSaL is roughly 40% of the ones for kernel/standard k-means clustering and MC-KUSaL is much better than SSC (with 32% reduction) in these experiments. In addition, the clustering error for rMC-KUSaL is an increasing function of the number of missing entries for both Gaussian kernel and polynomial kernel.

As another example, we repeat the above experiments using digits "1" and "6", where we again apply MC-KUSaL and rMC-KUSaL using Gaussian kernel $\kappa(\mathbf{y},\mathbf{y}') = \exp(-\frac{\|\mathbf{y}-\mathbf{y}'\|_2^2}{4})$ with $s = 35$ and polynomial kernel $\kappa(\mathbf{y},\mathbf{y}') = (\langle \mathbf{y},\mathbf{y}' \rangle + 1)^3$ with $s = 40$. In these experiments, SSC is performed with $\alpha_z = 10$. From Table 3.1, we again observe that MC-KUSaL outperforms other clustering algorithms with 42% reduction (compared to SSC) in the clustering error. In the missing data experiments, the clustering performance of rMC-KUSaL using Gaussian kernel degrades as the number of missing entries of the data increases. When we use polynomial kernel for rMC-KUSaL, increasing the number of entries in the missing data does not result in much degradation of the clustering performance.

We conclude by noting that the choice of kernels in these experiments is agnostic to the training data. Nonetheless, data-driven learning of kernels is an active area of research, which is sometimes studied under the rubric of *multiple kernel learning* [72–75]. While some of these works can be leveraged to further improve the performance of our proposed algorithms, a careful investigation of this is beyond the scope of this work.

Chapter 4

Hierarchical Union-of-Subspaces Model for Human Activity Summarization

4.1 Motivation

The need for high-level analytics of large streams of video data has arisen in recent years in many practical commercial, law enforcement, and military applications [38, 76–78]. Examples include human activity recognition, video summarization and indexing, human-machine teaming, and human behavior tracking and monitoring. The human activity recognition process requires recognizing both objects in the scene as well as body movements to correctly identify the activity with the help of context [79]. In the case of video summarization, one should be able to semantically segment and summarize the visual content in terms of context. This enables efficient indexing of large amounts of video data, which allows for easy query and retrieval [80]. For effective human-machine (robot) teaming, autonomous systems should be able to understand human teammates' gestures as well as recognize various human activities taking place in the field to gain situational awareness of the scene. Further, an important step in building visual tracking systems is to design ontologies and vocabularies for human activity and environment representations [81, 82]. All these tasks necessitate automated learning of movements of the human body or human action attributes.

Human activities consist of a sequence of actions that can be represented hierarchically [38] as shown in Fig. 4.1. The bottom level of the hierarchy consists of the fine resolution description of an action, i.e., movement of the human body (e.g., right arm moves up, left arm moves up, torso bending, and legs moving apart) and can be called an action attribute [39]. At the middle level, a sequence of these attributes forms a

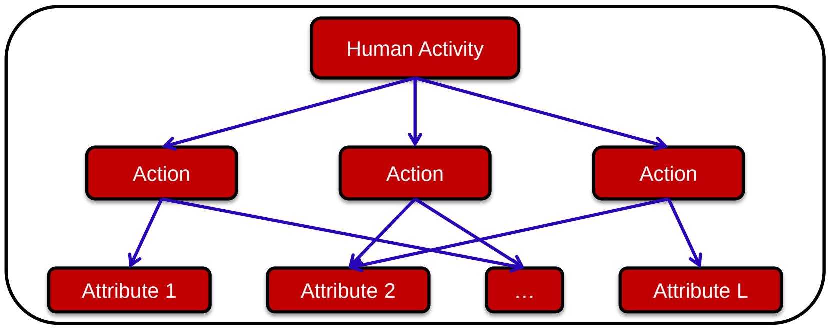


Figure 4.1: The hierarchical model for complex human activity representation.

human action. The human actions and their interactions form an activity, while a sequence of activities forms an event. An important advantage of the hierarchical model is that such structures go hand in hand with semantic or syntactic approaches and they provide us with the flexibility to generate summaries of long video sequences at different resolutions based on the needs of the end application.

In this work, we focus on the bottom two layers of the human activity hierarchical model, i.e., human actions and their representations using attributes. One possible way of obtaining such representations is to manually specify the action attributes and assign training video sequences to each attribute [39]. Another way is to manually annotate training video sequences by labeling movements [83]. Any human action in a test sequence can then be described using such user-defined attributes. Both of these approaches fall into supervised category. However, a set of user-defined action attributes may not completely describe all the human actions in given data. Also, manual assignment of training data for each of the action attributes is time consuming, if not impossible, for large datasets. Another issue with supervised learning of action attributes is that there might be attributes that were not seen in the training data by the system, but that might be seen in the field. Because of these reasons, unsupervised techniques to learn action attributes have been investigated [41,84,85]. These methods learn action attributes by clustering low-level features based on their co-occurrence in training videos. However, video data are not usually well distributed around cluster centers and hence, the cluster statistics may not be sufficient to accurately represent the attributes [13].

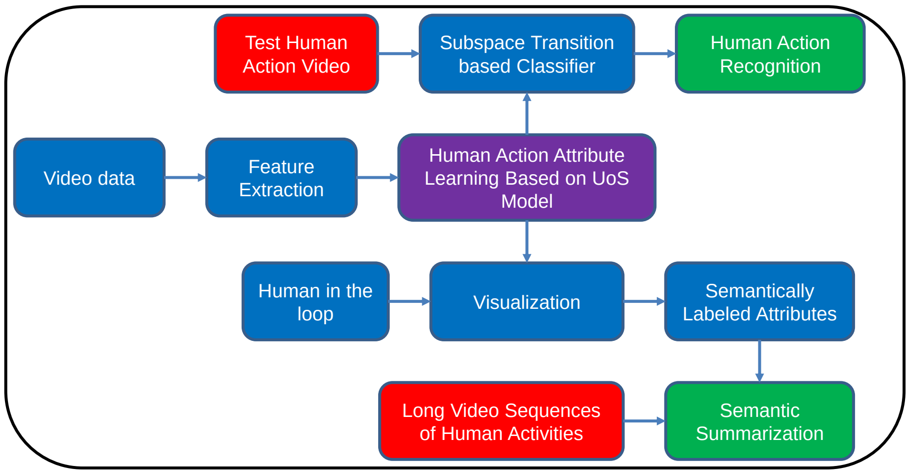

Test Human Action Video
Subspace Transition based Classifier
Human Action Recognition
Video data
Feature Extraction
Human Action Attribute Learning Based on UoS Model
Human in the loop
Visualization
Semantically Labeled Attributes
Long Video Sequences of Human Activities
Semantic Summarization

Figure 4.2: Block diagram of the human action recognition and summarization system.

Motivated by the premise that high-dimensional video data usually lie in a union of low-dimensional subspaces, instead of being uniformly distributed in the high-dimensional ambient space [13], we propose to represent human action attributes based on the *union-of-subspaces* (UoS) model [42]. The hypothesis of the UoS model is that each action attribute can be represented by a subspace. We conjecture that the action attributes represented by subspaces can encode more variations within an attribute compared to the representations obtained using co-occurrence statistics [41, 84, 85]. The task of unsupervised learning of the UoS underlying data of interest is often termed *subspace clustering* [13,42,86], which involves learning a graph associated with the data and then applying spectral clustering on the graph to infer the clustering of data.

The block diagram of the system that learns human action attributes is shown in Fig. 4.2. A large stream of video data is taken as input and features such as silhouettes, frame-by-frame spatial features like histograms of oriented gradients (HOG) [87], and spatio-temporal features like motion boundary histogram (MBH) [88] are extracted from the input. The data samples in this high-dimensional feature space are given as the input to the learning algorithm and human action attributes are obtained as the output. One of the main applications of learning the attributes based on the UoS model is *semantic summarization* of long video sequences. The attributes at different levels of the hierarchy can be labeled by an expert-in-the-loop by visualizing the first few basis vectors of each attribute (subspace) in the form of images. Once the labeled attributes are available, any long video sequence of human activity can then be semantically summarized at different levels of granularity based on the requirements of an application. Another major application of learning the attributes is *human action recognition*. A human action or activity can be represented as a sequence of transitions from one attribute to another, and hence, can be represented by a subspace transition vector. Even though multiple actions can share action attributes, each action or activity can be uniquely represented by its subspace transition vector. A classifier can be trained based upon these transition vectors to classify an action in a test video sequence into one of the actions in the training data.

In this chapter, we propose a hierarchical union-of-subspaces (UoS) model to learn human action attributes from the data at different resolutions in an unsupervised manner. We use the silhouette structure of the human (after background suppression and thresholding) as the feature in our approach. Each action attribute is represented by a subspace built from the silhouette features. We use Sparse Subspace Clustering (SSC) [13], a state-of-the-art subspace clustering method, as the basic subspaces learning algorithm and build our hierarchical UoS learning algorithm on top of it. Note that in the remainder of this chapter, we use the words "subspace" and "action attribute" interchangeably for convenience.

4.2 Background: Sparse Subspace Clustering

We start with a brief review of the Sparse Subspace Clustering (SSC) algorithm described in [13]. Suppose we are given a collection of N signals in $\mathbb{R}^m$, denoted by $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N] \in \mathbb{R}^{m \times N}$, and assume these N samples are drawn from a union of L subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^L$ of dimensions $\{d_\ell\}_{\ell=1}^L$ in $\mathbb{R}^m$, where every signal belongs to one of the subspaces in $\{\mathcal{S}_\ell\}_{\ell=1}^L$. Therefore, we can write $\mathbf{Y} = \mathbf{Y}_1 \cup \mathbf{Y}_2 \cup \cdots \cup \mathbf{Y}_L$ where each $\mathbf{Y}_\ell \in \mathbb{R}^{m \times N_\ell}$ is a submatrix of $\mathbf{Y}$ containing all the samples that belong to subspace $\mathcal{S}_\ell$ with $N_\ell > d_\ell$, and we have $\sum_{\ell=1}^L N_\ell = N$. Based on the intuition that each sample $\mathbf{y}_i$ can be expressed as a sparse linear combination of the data points from the same subspace to which $\mathbf{y}_i$ belongs, one can represent $\mathbf{y}_i$ as follows:

$$\widehat{\mathbf{a}}_i = \arg\min_{\mathbf{a}_i} \|\mathbf{a}_i\|_1 \quad \text{s.t.} \quad \mathbf{y}_i = \mathbf{Y}\mathbf{a}_i,\ a_{i_{(i)}} = 0, \tag{4.1}$$

where $\mathbf{a}_i = [a_{i_{(1)}}, a_{i_{(2)}}, \ldots, a_{i_{(N)}}]^T \in \mathbb{R}^N$ is the coefficient vector and $\|\mathbf{a}_i\|_1 = \sum_{j=1}^{N} |a_{i_{(j)}}|$. Considering all the data points in matrix form, SSC learns a sparse coefficient matrix $\widehat{\mathbf{A}} = [\widehat{\mathbf{a}}_1, \widehat{\mathbf{a}}_2, \ldots, \widehat{\mathbf{a}}_N] \in \mathbb{R}^{N\times N}$ by minimizing the following objective function:

$$\widehat{\mathbf{A}} = \arg\min_{\mathbf{A}} \|\mathbf{A}\|_1 \quad \text{s.t.} \quad \mathbf{Y} = \mathbf{Y}\mathbf{A},\ \text{diag}(\mathbf{A}) = \mathbf{0}, \tag{4.2}$$

where diag($\mathbf{A}$) is the diagonal vector of matrix $\mathbf{A}$. Using the resulting coefficient matrix $\widehat{\mathbf{A}}$, the segmentation of data points into respective subspaces $\mathbf{Y}_1, \ldots, \mathbf{Y}_L$ can be done by applying spectral clustering [71] on the similarity matrix $\mathbf{W} = |\widehat{\mathbf{A}}| + |\widehat{\mathbf{A}}|^T$, where $|\cdot|$ denotes the element-wise absolute value operation. In the case when data are corrupted by noise, SSC solves the following convex optimization problem for $\widehat{\mathbf{A}}$:

$$\widehat{\mathbf{A}} = \arg\min_{\mathbf{A}} \|\mathbf{A}\|_1 + \lambda\|\mathbf{Y} - \mathbf{Y}\mathbf{A}\|_F^2 \quad \text{s.t.} \quad \text{diag}(\mathbf{A}) = \mathbf{0}. \tag{4.3}$$

Here, $\lambda > 0$ is a regularization parameter. The authors in [13] proposed an efficient solution for calculating the sparse coefficients $\widehat{\mathbf{A}}$ using Alternating Direction Method of Multipliers (ADMM) [89]. Note that there also exist some other UoS learning approaches, such as robust sparse subspace clustering (RSSC) [34] and robust subspace clustering via thresholding (TSC) [33]. In this work, we propose our hierarchical UoS learning algorithm based on SSC due to its superior performance, although our approach is extendable to other algorithms.

4.3 Hierarchical Sparse Subspace Clustering

In this section, we introduce our Hierarchical Sparse Subspace Clustering (HSSC) algorithm for learning multiple levels of UoS using a collection of high-dimensional data. We use $\mathbf{Y}_{p|\ell} \in \mathbb{R}^{m\times N_{p|\ell}}$ to denote the set of signals that are assigned to the ℓ-th subspace at the p-th level of the hierarchical structure, where $N_{p|\ell}$ is the number of signals in $\mathbf{Y}_{p|\ell}$. Let L_p denote the number of subspaces at the p-th level, then we have $\sum_{\ell=1}^{L_p} N_{p|\ell} = N$ and $\mathbf{Y} = \bigcup_{\ell=1}^{L_p} \mathbf{Y}_{p|\ell}$ for all p's. The subspace underlying $\mathbf{Y}_{p|\ell}$ is denoted by $\mathcal{S}_{p|\ell}$ and its orthonormal basis is denoted by $\mathbf{D}_{p|\ell} \in \mathbb{R}^{m \times d_{p|\ell}}$, where $d_{p|\ell}$ denotes the dimension of the subspace $\mathcal{S}_{p|\ell}$.

We begin by applying SSC on $\mathbf{Y}$ at the first level ($p = 1$), which divides $\mathbf{Y}$ into two subspaces with clusters $\mathbf{Y} = \mathbf{Y}_{1|1} \cup \mathbf{Y}_{1|2}$ such that $\mathbf{Y}_{1|1} \cap \mathbf{Y}_{1|2} = \emptyset$. At the second level, we again perform SSC on $\mathbf{Y}_{1|1}$ and $\mathbf{Y}_{1|2}$ separately and divide each of them into 2 clusters, yielding 4 clusters $\mathbf{Y} = \bigcup_{\ell=1}^{4} \mathbf{Y}_{2|\ell}$ with $\mathbf{Y}_{1|\ell} = \mathbf{Y}_{2|2\ell-1} \cup \mathbf{Y}_{2|2\ell}$ ($\ell = 1, 2$). Using the signals in $\mathbf{Y}_{2|\ell}$ ($\ell = 1, \ldots, 4$), we estimate the four subspaces $\mathcal{S}_{2|\ell}$'s underlying $\mathbf{Y}_{2|\ell}$'s by identifying their orthonormal bases $\mathbf{D}_{2|\ell}$'s. To be specific, we obtain eigendecomposition of the covariance matrix $\mathbf{C}_{2|\ell} = \mathbf{Y}_{2|\ell}\mathbf{Y}_{2|\ell}^T$ as $\mathbf{C}_{2|\ell} = \mathbf{U}_{2|\ell}\mathbf{\Sigma}_{2|\ell}\mathbf{U}_{2|\ell}^T$, where $\mathbf{\Sigma}_{2|\ell} = \text{diag}(\lambda_1, \ldots, \lambda_{N_{2|\ell}})$ is a diagonal matrix ($\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{N_{2|\ell}}$) and $\mathbf{U}_{2|\ell} = [\mathbf{u}_1, \ldots, \mathbf{u}_{N_{2|\ell}}]$. Then the dimension of the subspace $\mathcal{S}_{2|\ell}$, denoted by $d_{2|\ell}$, is estimated based on the energy threshold, i.e., $d_{2|\ell} = \arg\min_d \frac{\sum_{j=1}^{d} \lambda_j}{\sum_{j=1}^{N_{2|\ell}} \lambda_j} \geq \alpha$, where α is a predefined threshold and is set close to 1 for better representation. The orthonormal basis of $\mathcal{S}_{2|\ell}$ can then be written as $\mathbf{D}_{2|\ell} = [\mathbf{u}_1, \ldots, \mathbf{u}_{d_{2|\ell}}]$. After this step, we end up with 4 subspaces with clusters $\{\mathbf{Y}_{2|\ell}\}_{\ell=1}^{4}$ and their associated orthonormal bases $\{\mathbf{D}_{2|\ell}\}_{\ell=1}^{4}$.

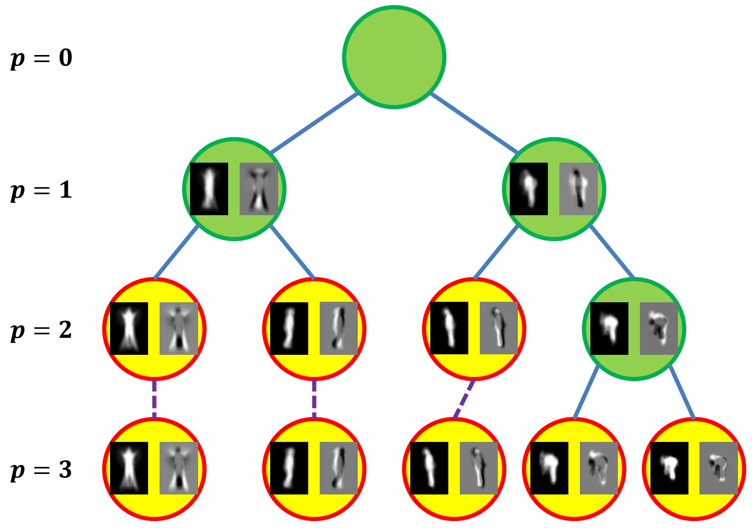


Figure 4.3: An example of using HSSC to learn a hierarchical UoS model. Each circle represents one cluster/subspace, and the first two vectors of each $\mathbf{D}_{p,\ell}$ are plotted in each circle. The green nodes represent the clusters that are further divided in the next level of the hierarchy. Leaf nodes are represented as yellow, these clusters cannot be further divided and are the final attributes obtained at the bottom most level of the hierarchical model.

Algorithm 6: Hierarchical Sparse Subspace Clustering (the p-th level)

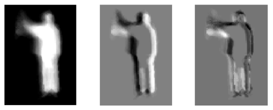
```
Input: A set of clusters $\{\mathbf{Y}_{p|\ell}\}_{\ell=1}^{L_p}$, their underlying subspace bases $\{\mathbf{D}_{p|\ell}\}_{\ell=1}^{L_p}$
  and $\{g_{p|\ell}\}_{\ell=1}^{L_p}$, and parameters $\alpha$, $\beta$ and $d_{\min}$.
 1: $\widehat{L} = 0$.
 2: for all $\ell = 1$ to $L_p$ do
 3:   if $g_{p|\ell} = 1$ then
 4:     $\theta = 0$.
 5:     Split $\mathbf{Y}_{p|\ell}$ into $\mathbf{Z}_1$ and $\mathbf{Z}_2$ using SSC.
 6:     $\forall k = 1, 2$, estimate $d_{\mathbf{Z}_k}$ and $\mathbf{D}_{\mathbf{Z}_k}$ of $\mathcal{S}_{\mathbf{Z}_k}$ using $\mathbf{Z}_k$.
 7:     $\forall k = 1, 2$, compute $E_k$ and $\widehat{E}_k$.
        If $(E_k - \widehat{E}_k)/E_k \geq \beta$, $\theta = \theta + 1$.
 8:     If $\theta \geq 1$ and $\min(d_{\mathbf{Z}_1}, d_{\mathbf{Z}_2}) \geq d_{\min}$
 9:        $\forall k = 1, 2$, $\mathbf{Y}_{p+1|\widehat{L}+k} = \mathbf{Z}_k$, $\mathbf{D}_{p+1|\widehat{L}+k} = \mathbf{D}_{\mathbf{Z}_k}$, and $g_{p+1|\widehat{L}+k} = 1$.
10:        $\widehat{L} = \widehat{L} + 2$.
11:     Else $\mathbf{Y}_{p+1|\widehat{L}+1} = \mathbf{Y}_{p|\ell}$, $\mathbf{D}_{p+1|\widehat{L}+1} = \mathbf{D}_{p|\ell}$, $g_{p+1|\widehat{L}+1} = 0$, and $\widehat{L} = \widehat{L} + 1$.
12:   else
13:     $\mathbf{Y}_{p+1|\widehat{L}+1} = \mathbf{Y}_{p|\ell}$, $\mathbf{D}_{p+1|\widehat{L}+1} = \mathbf{D}_{p|\ell}$, $g_{p+1|\widehat{L}+1} = 0$, and $\widehat{L} = \widehat{L} + 1$.
14:   end if
15: end for
16: $L_{p+1} = \widehat{L}$.
Output: A set of clusters $\{\mathbf{Y}_{p+1|\ell}\}_{\ell=1}^{L_{p+1}}$, subspace bases $\{\mathbf{D}_{p+1|\ell}\}_{\ell=1}^{L_{p+1}}$ and
  $\{g_{p+1|\ell}\}_{\ell=1}^{L_{p+1}}$.
```

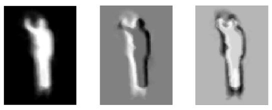
For every $p \geq 2$, we decide whether or not to further divide each single cluster or subspace at the p-th level into two clusters or subspaces at the $(p+1)$-th level based on the following principle. We use a binary variable $g_{p|\ell}$ to indicate whether the cluster $\mathbf{Y}_{p|\ell}$ is further divisible at the next level or not. If it is, we set $g_{p|\ell} = 1$, otherwise $g_{p|\ell} = 0$. We initialize $g_{2|\ell} = 1$ for all ℓ's ($\ell = 1, \ldots, 4$). Consider the cluster $\mathbf{Y}_{p|\ell}$ at the p-th level and assume that $\widehat{L}$ clusters already exist at the $(p+1)$-th level derived from $\{\mathbf{Y}_{p|1}, \mathbf{Y}_{p|2}, \ldots, \mathbf{Y}_{p|\ell-1}\}$. If $g_{p|\ell} = 0$, the $(\widehat{L}+1)$-th cluster at the $(p+1)$-th level will be the same as $\mathbf{Y}_{p|\ell}$; thus, we simply set $\mathbf{Y}_{p+1|\widehat{L}+1} = \mathbf{Y}_{p|\ell}$, $\mathbf{D}_{p+1|\widehat{L}+1} = \mathbf{D}_{p|\ell}$ and $g_{p+1|\widehat{L}+1} = 0$. If $g_{p|\ell} = 1$ (in which case $\mathbf{Y}_{p|\ell}$ corresponds to the green nodes in Fig. 4.3), we first split $\mathbf{Y}_{p|\ell}$ into two sub-clusters $\mathbf{Y}_{p|\ell} = \mathbf{Z}_1 \cup \mathbf{Z}_2$ using SSC and find the best subspaces $\mathcal{S}_{\mathbf{Z}_k}$ ($k = 1, 2$) that fit $\mathbf{Z}_k$'s respectively using the aforementioned strategy, while their dimensions and orthonormal bases are denoted by $d_{\mathbf{Z}_k}$'s and $\mathbf{D}_{\mathbf{Z}_k}$'s, respectively. Then for every signal $\mathbf{y}_i$ in $\mathbf{Z}_k$ ($k = 1, 2$), we compute the relative representation error of $\mathbf{y}_i$ using the parent subspace basis $\mathbf{D}_{p|\ell}$ and the child subspace basis $\mathbf{D}_{\mathbf{Z}_k}$, which are defined as $e_i = \frac{\|\mathbf{y}_i - \mathbf{D}_{p|\ell}\mathbf{D}_{p|\ell}^T\mathbf{y}_i\|_2^2}{\|\mathbf{y}_i\|_2^2}$ and $\widehat{e}_i = \frac{\|\mathbf{y}_i - \mathbf{D}_{\mathbf{Z}_k}\mathbf{D}_{\mathbf{Z}_k}^T\mathbf{y}_i\|_2^2}{\|\mathbf{y}_i\|_2^2}$, respectively. The means of the relative reconstruction errors of all the signals in $\mathbf{Z}_k$ using $\mathbf{D}_{p|\ell}$ and $\mathbf{D}_{\mathbf{Z}_k}$ are denoted by E_k and $\widehat{E}_k$, respectively. Finally, we divide $\mathbf{Y}_{p|\ell}$ into $\mathbf{Z}_1 \cup \mathbf{Z}_2$ if (i) the relative reconstruction errors of the signals using the child subspace are less than the reconstruction errors of the signals using the parent subspace by a certain threshold, i.e., $(E_k - \widehat{E}_k)/E_k \geq \beta$ for either $k = 1$ or 2, and (ii) the dimensions of the two child subspaces meet a minimum requirement, that is, $\min(d_{\mathbf{Z}_1}, d_{\mathbf{Z}_2}) \geq d_{\min}$. In here, β and $d_{\min}$ are user-defined parameters and are set to avoid redundant subspaces. When either β or $d_{\min}$ decreases, we will have more subspaces. Assuming the two conditions are satisfied, the cluster $\mathbf{Y}_{p|\ell}$ is then divided by setting $\mathbf{Y}_{p+1|\widehat{L}+1} = \mathbf{Z}_1$ $(g_{p+1|\widehat{L}+1} = 1)$ and $\mathbf{Y}_{p+1|\widehat{L}+2} = \mathbf{Z}_2$ $(g_{p+1|\widehat{L}+2} = 1)$. The bases of the subspaces at the $(p+1)$-th level are set by $\mathbf{D}_{p+1|\widehat{L}+1} = \mathbf{D}_{\mathbf{Z}_1}$ and $\mathbf{D}_{p+1|\widehat{L}+2} = \mathbf{D}_{\mathbf{Z}_2}$. If the above conditions are not satisfied, we set $\mathbf{Y}_{p+1|\widehat{L}+1} = \mathbf{Y}_{p|\ell}$, $g_{p|\ell} = 0$ and $g_{p+1|\widehat{L}+1} = 0$ to indicate $\mathbf{Y}_{p|\ell}$, i.e., $\mathbf{Y}_{p+1|\widehat{L}+1}$, is a leaf cluster and this cluster will not be divided any further (which corresponds to the yellow nodes in Fig. 4.3). This process is repeated until we reach a predefined maximum level in the hierarchy denoted by P. The hierarchical SSC algorithm for any level $p \geq 2$ is described in Algorithm 6.

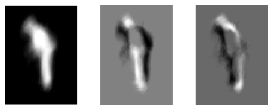
Fig. 4.3 also shows an example of applying HSSC to determine the action attributes for three actions: bend, jumping jack and jump in the Weizmann dataset [90]. Here, the maximum number of levels in the model P is set to 3 because we don't expect to have more than $2^3 = 8$ subspaces at the bottom level for only three actions. The hierarchical UoS model is initialized with the silhouette features of all the actions from multiple subjects at the top $(p = 0)$. Each silhouette frame (obtained after background suppression and thresholding) in the video sequence is a data sample $\mathbf{y}_i$ in both SSC and HSSC. Inside each node, we visualize the first two basis vectors of the subspaces obtained at each level. We can see that at the first level $(p = 1)$, the attributes corresponding to two actions, jumping jack and jump are represented by one subspace and attributes corresponding to the bend action are represented by another subspace. At the second level $(p = 2)$, the action attributes corresponding to jumping jack and jump are separated into two different subspaces. While the action attributes of the bend action, which is a more complex action with wider range of movement, are divided into two subspaces, representing the more upright part of the bend action as one higher resolution attribute and the lower part of the bend action as another higher resolution attribute. The lower part of the bend action is further divided into two more attributes at the next level ($p = 3$) while the other attributes are left as they were. Thus, we can see that as p increases, the variations within each action can be identified, extracted and represented using more number of higher resolution action attributes.

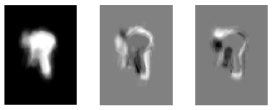
The proposed HSSC algorithm has some obvious advantages over flat SSC for learning human action attributes. First, in the case of flat SSC, one has to define the number of subspaces into which the data are to be clustered [13]. This requirement puts a constraint of prior knowledge about the data in that we need to know the number of underlying human action attributes present in the data. Such an approach moves away from data-driven learning. On the other hand, HSSC algorithm only requires the knowledge of a maximum level P, and it can stop before it reaches the P-th level if no clusters can be further divided. HSSC algorithm is designed in such a way that all the variations within each action can be identified automatically to determine the final number of action attributes. Second, HSSC provides us with multiple resolutions of action attributes, which are extremely useful for semantic labeling and understanding of human actions. The multi-resolution attributes can be used for semantic summarization of long video sequences at different resolutions starting from giving just an overview of the action to detailed explanation of movements occurring in the video. Flat SSC can provide us with only one set of action attributes at one single resolution, which depends on the number of subspaces that is the input to the algorithm. The empirical results presented in Section 4.4 illustrate these benefits of HSSC with examples.

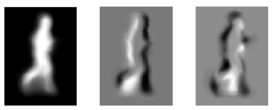
To demonstrate the reason why HSSC outperforms flat SSC, we consider the following example. We perform HSSC with $P = 3$ to learn attributes for three actions: bend, run and one-hand wave in Weizmann dataset [90]. It has 6 subspaces as the leaf nodes, and the first three basis vectors of each leaf subspace are illustrated in Fig. 4.4 (a)-(f). We then apply SSC to learn 6 attributes and all those subspaces are depicted in Fig. 4.4

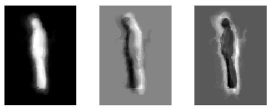

(a) Stand with one hand out

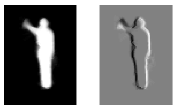

(b) Stand with one hand up

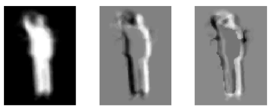

(c) 45 degree bend

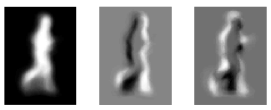

(d) 90 degree bend

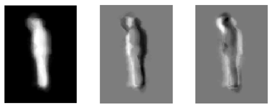

(e) Run

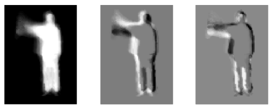

(f) 30 degree bend


(g) Stand with one hand out


(h) Stand with one hand up


(i) Run


(j) 30 degree bend


(k) Stand with one hand out

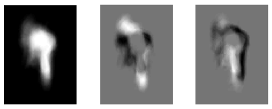
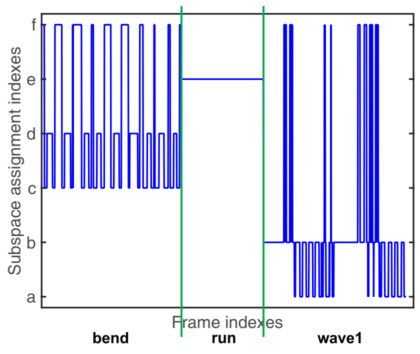
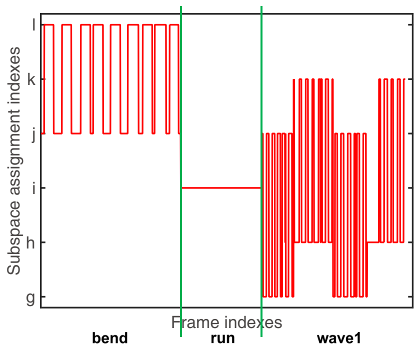

(l) Bend

Figure 4.4: Visualization of subspaces (first three dimensions) learned using the frames of three actions: bend, run and one-hand wave. (a)-(f) represent the leaf subspaces learned by HSSC. (g)-(l) represent the subspaces which are learned using SSC.

(g)-(l). It can be seen that the attributes learned using hierarchical SSC capture the variations and full range of movement within the bend action in a better way compared to SSC (as can be seen in (c), (d) (f) and (j), (l)). While the first subspace of SSC (seen in (g)) does not provide any additional information of one-hand wave action. We also present the three action video sequences for 9 subjects in Weizmann dataset as transition sequences of the learned attributes in Fig. 4.5. The run action is represented by a single subspace in both methods. As expected, the bend action is represented by more number of attributes in HSSC compared to SSC. In terms of subspace transitions, some frames of one-hand wave action are represented by attribute (f) in HSSC, which corresponds to the bend action attribute. However, in SSC, there are more frames in one-hand wave action which are represented by the attribute (j) (see Fig. 4.5(b)). Thus, the action recognition performance using HSSC will be better compared to SSC.



(a)



(b)

Figure 4.5: Subspace assignment result of the video frames from bend, run and one-hand wave actions using subspaces learned from (a) HSSC and (b) SSC.

4.3.1 Complexity Analysis for Flat SSC and Hierarchical SSC

We first investigate the computational complexity of flat SSC. As proposed in [13], SSC mainly consists of three steps: learning the sparse coefficients $\widehat{\mathbf{A}}$ using ADMM, computing the normalized Laplacian matrix from $\widehat{\mathbf{A}}$, and k-means clustering (with L clusters) on the normalized Laplacian matrix. For the first two steps, the complexity is implementation dependent, but the worst case would be $\mathcal{O}(N^3)$, where N is the number of samples in the training data $\mathbf{Y}$. The complexity of k-means clustering on the normalized Laplacian matrix is $\mathcal{O}(N^2L)$. Therefore, by assuming $L \ll N$, the overall complexity of SSC is $\mathcal{O}(N^3)$.

To analyze the computational complexity of hierarchical SSC, we assume that there are 2^p clusters at the p-th level ($p = 0, 1, \ldots, \log_2 L$). For each cluster at the p-th level ($0 \leq p \leq \log_2 L - 1$), we run SSC on each cluster to obtain two sub-clusters at the next level. We again use $N_{p|\ell}$ to denote number of signals that are assigned to the ℓ-th cluster at the p-th level. As discussed earlier, the computational complexity of applying SSC on these $N_{p|\ell}$ samples for two clusters will be $\mathcal{O}(N_{p|\ell}^3)$. For the sake of exposition, we make another assumption that $N_{p|\ell} = N/2^p$ for all ℓ's. In such a case, the overall complexity at the p-th level is $\mathcal{O}((N/2^p)^3 \times 2^p) = \mathcal{O}(N^3/4^p)$. The sum of the complexity orders over all the levels gives us the overall complexity of hierarchical SSC as $\sum_{p=0}^{\log_2 L-1} \mathcal{O}(N^3/4^p) = \mathcal{O}(\frac{4}{3}N^3(1 - \frac{1}{L^2}))$. The computational complexity of HSSC is slightly more than that of flat SSC. However, the advantages of HSSC over flat SSC in learning better action attributes at different resolutions without any prior knowledge of the number of attributes significantly outweighs this slight increase in computational complexity.

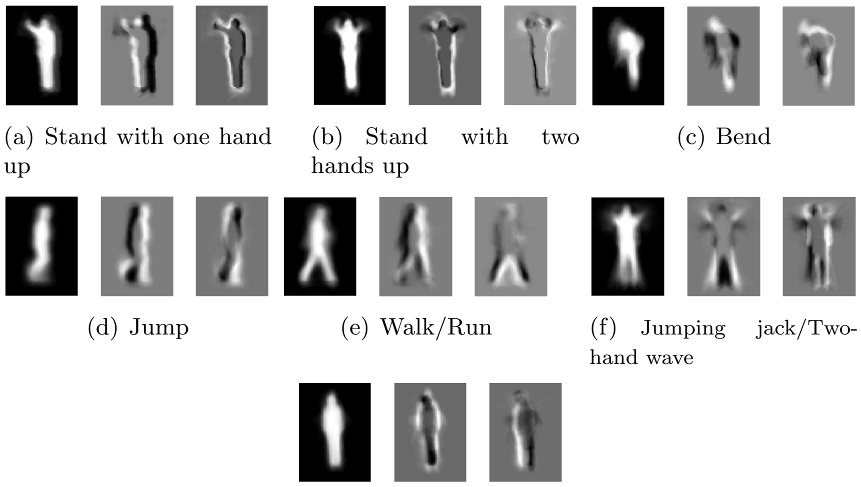
4.3.2 Action Recognition Using Learned Subspaces

In this section, we describe the classification strategy to perform action recognition using the hierarchical UoS model learned by HSSC algorithm. We first learn the subspaces, which are the action attributes, by applying HSSC on the silhouette feature sequences of human actions in an unsupervised manner, where each silhouette frame (each data sample) is vectorized and normalized to unit ℓ_2 norm. We assume HSSC ends up with L_P leaf subspaces and the orthonormal bases of these subspaces can be represented by $\{\mathbf{D}_{P|\ell} \in \mathbb{R}^{m \times d_{P|\ell}}\}_{\ell=1}^{L_P}$.

Suppose there are V actions and R subjects in the training set. We use $\mathbf{\Phi}_{v,r} \in \mathbb{R}^{m \times s_{v,r}}$ to denote the video sequence of the r-th subject with the v-th action, where $s_{v,r}$ denotes the number of frames in the video. We assign every frame in one video sequence $\mathbf{\Phi}_{v,r}$ ($v \in \{1, \ldots, V\}, r \in \{1, \ldots, R\}$) to the "closest leaf subspace" and we use $\boldsymbol{\phi}_{v,r} \in \mathbb{R}^{s_{v,r}}$ to denote the vector which contains the resulting subspace assignment indexes. This vector represents the sequence of action attributes and the transitions involved in the human action video. All training video samples have subspace transition vectors $\boldsymbol{\phi}_{v,r}$'s. Then for a test video $\mathbf{\Psi} \in \mathbb{R}^{m \times s}$ with s denoting the number of frames in this video, we first perform subspace assignment for all the frames in $\mathbf{\Psi}$ and we use $\boldsymbol{\psi} \in \mathbb{R}^{s}$ to denote the resulting transition vector. Then we use a nearest neighbor classifier to perform action recognition, i.e., $\mathbf{\Psi}$ is declared to be in a particular action class v' for which the average of distances between the transition vector $\boldsymbol{\psi}$ and all the training transition vectors $\boldsymbol{\phi}_{v',r}$'s in the v'-th class is the smallest. Note that the video sequences, and hence the subspace assignment vectors, are of different lengths. In order to make the action recognition process temporal-scale invariant, we use Dynamic Time Warping (DTW) method [91] on the Grassmann manifold, described in Algorithm 7, where the element-wise distances used in here are the normalized subspace distances between the leaf subspaces. Mathematically speaking, for every pair of the subspaces $\mathcal{S}_{P|\ell}$ and $\mathcal{S}_{P|\widehat{\ell}}$, the normalized subspace distance between these two subspaces on the Grassmann manifold (in Step 3 of Algorithm 7) is defined as $d_u(\mathcal{S}_{P|\ell}, \mathcal{S}_{P|\widehat{\ell}}) = \sqrt{1 - \frac{\mathrm{tr}(\mathbf{D}_{P|\ell}^T \mathbf{D}_{P|\widehat{\ell}} \mathbf{D}_{P|\widehat{\ell}}^T \mathbf{D}_{P|\ell})}{\max(d_{P|\ell}, d_{P|\widehat{\ell}})}}$ [44].

Algorithm 7: Dynamic Time Warping on the Grassmann Manifold

```
Input: Two subspace assignment sequences $\boldsymbol{\phi} \in \mathbb{R}^{s_1}$ and $\boldsymbol{\psi} \in \mathbb{R}^{s_2}$, leaf subspace
  bases $\{\mathbf{D}_{P|\ell}\}_{\ell=1}^{L_P}$ of $\mathcal{S}_{P|\ell}$'s.
Initialize: A matrix $\mathbf{E} \in \mathbb{R}^{(s_1+1)\times(s_2+1)}$ with $e_{1,1} = 0$ and all other entries in the
  first row and column are $\infty$.
1: for all $i = 1$ to $s_1$ do
2:   for all $j = 1$ to $s_2$ do
3:     $e_{i+1,j+1} = d_u(\mathcal{S}_{P|\phi_{(i)}}, \mathcal{S}_{P|\psi_{(j)}}) + \min(e_{i,j+1}, e_{i+1,j}, e_{i,j})$.
4:   end for
5: end for
Output: The distance between $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ is $e_{s_1+1,s_2+1}$.
```



(a) Stand with one hand up

(b) Stand with two hands up

(c) Bend

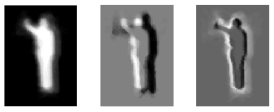
(d) Jump

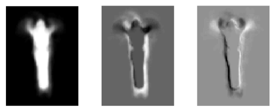
(e) Walk/Run

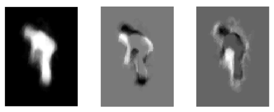
(f) Jumping jack/Two-hand wave

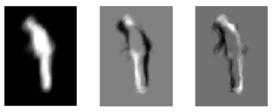
(g) Stand

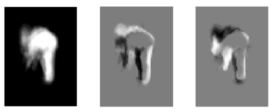
Figure 4.6: Visualization and interpretation of attributes at the 3rd level of HSSC for Weizmann dataset.

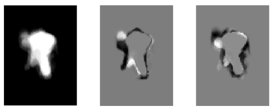

(a) One-hand wave

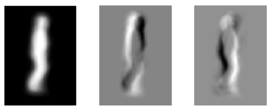

(b) Stand with two hands up

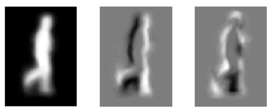

(c) 45 degree bend

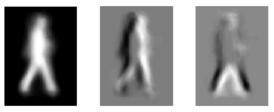

(d) 30 degree bend

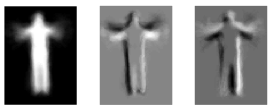

(e) 60 degree bend

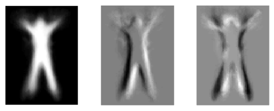

(f) 90 degree bend

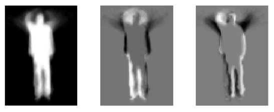

(g) Jump


(h) Run


(i) Walk


(j) Stand with two hands out


(k) Jumping jack


(l) Stand with two hands up

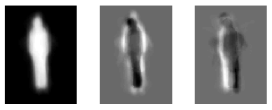

(m) Stand

Figure 4.7: Visualization and interpretation of attributes at the bottom (5th) level of HSSC for Weizmann dataset.

4.4 Performance Evaluation

In this section, we report the experimental results obtained by applying the proposed HSSC approach on human action video datasets to learn the action attributes. Our first objective is to semantically interpret and label the learned action attributes from HSSC and to investigate the utility of the multi-resolution action attributes in semantic description of long action sequences. The secondary goal is to evaluate the effectiveness of these learned attributes in human action recognition and to compare the quantitative results to other UoS learning methods. In all the following experiments, we use the noisy variant of the optimization program (i.e., ADMM) of SSC and set $\lambda_z = \alpha_z/\mu_z$, where λ_z and μ_z are as defined in [13, (13) & (14)] and the parameter α_z varies in different experiments.

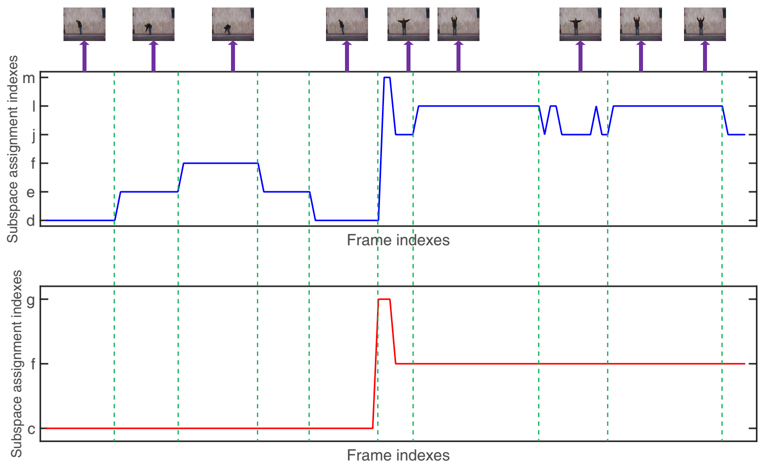


Figure 4.8: Subspace transition of a long sequence using subspaces at the 5th level (top) and the 3rd level (bottom). The subspace assignment indexes in the top/bottom figure correspond to the attributes in Fig. 4.7 and Fig. 4.6, respectively.

4.4.1 Semantic Labeling and Summarization

In this section, we visualize the learned attributes from HSSC at two different resolutions, give them semantic labels and use them for semantic summarization of multiple actions in a long video sequence. We apply HSSC on the Weizmann dataset with parameters $P = 5$, $\alpha = 0.9$, $\beta = 0.05$, $d_{\min} = 4$ and $\alpha_z = 20$. HSSC returns $L_P = 13$ leaf subspaces at the 5th level and 7 subspaces at the 3rd level. We show the first three dimensions of the orthonormal bases of those subspaces (attributes) here and give them interpretative (semantic) labels in Fig. 4.7 and Fig. 4.6, respectively. To demonstrate the semantic summarization of a long video sequence, we create a sequence by concatenating the bend and two-hand wave sequence of one subject and visualize the subspace transition of the frames in Fig. 4.8. We can interpret the actions using the attribute assignment within the interval defined by green lines based on the corresponding labels in Fig. 4.7 and Fig. 4.6. At Level 5 (Fig. 4.8, top), human actions in the first half of the video sequence can be described as 30 degree bend followed by 60 degree bend, 90 degree bend, 60 degree bend again, and 30 degree bend, which can be interpreted as a full range bend action as done in Level 3 (Fig. 4.8, bottom). Next, at Level 5, the actions in the second half of the video sequence are Stand followed by two alternating attributes: Stand with two hands out and Stand with two hands up. The complete action can be described as a two-hand wave, which is precisely what is done at a lower resolution in Level 3. Therefore, we can say that the attributes generated at different levels of HSSC algorithm can be used for semantic summarization of video sequences at different resolutions.

4.4.2 Action Recognition: Evaluation on Different Datasets

In this section, we compare the performance of the proposed HSSC algorithm to flat SSC [13], RSSC [34], and TSC [33] with the number of clusters set (*i*) to be the same number of subspaces generated by HSSC at the bottom-most level (which is denoted by *Algorithm*-L_P) and (*ii*) to be the same as the number of actions (which is denoted by *Algorithm*-V). The parameter α_z for flat SSC is the same as the one for HSSC. In the case of RSSC, we set $\lambda = 1/\sqrt{d}$ as per [34], where d is the mean of the subspace dimensions returned by SSC-L_P/SSC-V. The tuning parameter q in TSC is set $q = \max(3, \lceil N/(L \times 20) \rceil)$, where L is equal to L_P and V for TSC-L_P and TSC-V, respectively.

We use three public datasets for this purpose: the Weizmann action dataset [90], the Keck gesture dataset [92], and the UT-Tower action dataset [93]. We evaluate all the subspace/attribute learning approaches based on a leave-one-subject-out experiment. To be specific, we pick all the videos of one subject (see Section 4.3.2) for testing at one time, while using all other videos as training samples. The Weizmann dataset consists of $V = 10$ different actions: walk, run, jump, gallop sideways, bend, one-hand wave, two-hands wave, jump in place, jumping jack, and skip. Each action is performed by nine subjects. The original resolution of the frames is 180×144. We align all the binary silhouette sequences and crop them into 87×63 frames, thereby the dimensionality of data is $m = 5481$. The hierarchical SSC is performed with parameters described in Section 4.4.1 and it returns $L_P = 13$ leaf subspaces for final attributes. The Keck gesture dataset was collected using a camera with 640×480 resolution. It consists of $V = 14$ different actions, including turn left, turn right, attention left, attention right, flap, stop left, stop right, stop both, attention both, start, go back, close distance, speed up, and come near. Each of these 14 actions is performed by three people. In each video sequence, the subject repeats the same action three times. Therefore the total number of video sequences in this dataset is $14\times3\times3 = 126$. We crop all the silhouette sequences to 380×280 resolution and downsample all the video frames by a factor of 4 in each dimension for computational purposes, with the resulting sequences being of size 95×70 and hence $m = 6650$. We perform hierarchical SSC with parameters $P = 6$, $\alpha = 0.98$, $\beta = 0.02$, $d_{\min} = 3$ and $\alpha_z = 100$, in which case it returns $L_P = 18$ leaf subspaces at the bottom-most level. The UT-Tower action dataset contains a collection of 108 low resolution videos and there exist $V = 9$ different actions in this dataset, including pointing, standing, digging, walking, carrying, running, wave1, wave2, and jumping. Each action is performed twice by 6 individuals, which results in a total of 12 video sequences per action. We use the bounding boxes and foreground masks provided by the authors of [93] to extract silhouettes. All the silhouette sequences are of size 49×61 ($m = 2989$). We perform hierarchical SSC with parameters $P = 6$, $\alpha = 0.95$, $\beta = 0.04$, $d_{\min} = 4$ and $\alpha_z = 150$, obtaining $L_P = 11$ final subspaces. The recognition results of different algorithms for the three datasets are shown in Table 4.1. We can see that by representing the human actions using the attributes learned by HSSC, we are able to recognize the actions at a superior rate compared to other techniques.

Table 4.1: Recognition results (%) on different datasets

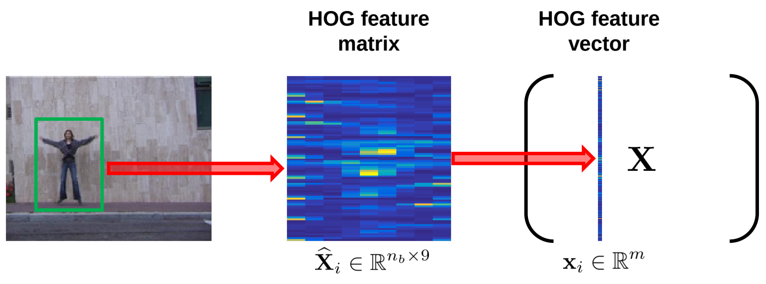
Data ↓ Method →	HSSC	SSC-L_P	SSC-V	RSSC-L_P	RSSC-V	TSC-L_P	TSC-V
Weizmann [90]	**91.11**	83.33	76.67	57.78	65.56	87.78	83.33
Keck [92]	**78.57**	57.94	67.46	34.13	37.30	53.17	53.97
UT-Tower [93]	**76.85**	75.93	73.15	60.19	63.89	65.74	62.04

Chapter 5

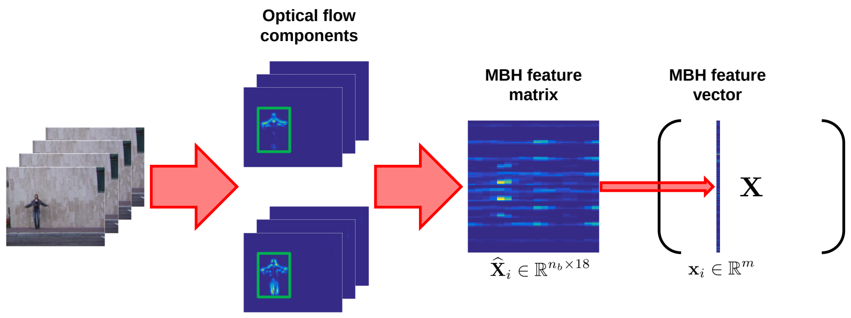
Human Action Attribute Learning Using Low-Rank Representations

In Chapter 4, we use the silhouette structure as the basic feature to learn action attributes. However, it is almost impossible to obtain clean silhouette features in many real applications due to cluttered background and drastic changes in the background. On the other hand, the ℓ_1 graph built from SSC does not consider the global structure of the data. Similar features may have drastically different coefficients for subspace clustering, which will degrade the clustering performance. To capture the global structure of data, low-rank representation (LRR) models with and without sparsity constraints have been proposed in [94] and [42], respectively. It has been proved that LRR can achieve perfect subspace clustering results under the condition that the subspaces underlying the data are independent [42, 95]. However, this condition is hard to satisfy in many real situations. To handle the case of disjoint subspaces, Tang *et al.* [96] extended LRR by imposing restrictions on the structure of the solution, called structure-constrained LRR (SC-LRR).

Existing LRR based subspace clustering techniques use spectral clustering as a post-processing step on the graph generated from a low-rank coefficient matrix, but the relationship between the coefficient matrix and the segmentation of data is seldom considered, which can lead to sub-optimal results [97]. In this chapter, we propose a hierarchical clustering-aware structure-constrained LRR (HCS-LRR) model, for unsupervised learning of human action attributes using other low-level features such as HOG [87] and MBH [88].


HOG feature matrix
HOG feature vector
X
$\widehat{\mathbf{X}}_i \in \mathbb{R}^{n_b \times 9}$
$\mathbf{x}_i \in \mathbb{R}^m$

(a)


Optical flow components
MBH feature matrix
MBH feature vector
X
$\widehat{\mathbf{X}}_i \in \mathbb{R}^{n_b \times 18}$
$\mathbf{x}_i \in \mathbb{R}^m$

(b)

Figure 5.1: Illustration of our approach to create the matrix $\mathbf{X}$ for (a) HOG and (b) MBH features, which is then input to the HCS-LRR algorithm.

5.1 Feature Extraction for Attribute Learning

The main focus of our work is to learn meaningful human action attributes from large streams of video data in an unsupervised manner. The first step in our proposed framework is to extract feature descriptors from an action interest region in which the human performs the action. The action interest region of each frame of an action sequence is determined by a bounding box. In our work, we learn action attributes using two local visual descriptors: HOG (histograms of oriented gradients) [87] and MBH (motion boundary histogram) [88]. To extract HOG descriptors, we divide the action interest region into a grid of blocks, each of size $n_\sigma \times n_\sigma$ pixels. Then HOG feature is extracted for each block and orientations are quantized into 9 bins. Therefore, the HOG feature of every frame can be stored into a matrix $\widehat{\mathbf{X}}_i \in \mathbb{R}^{n_b \times 9}$, where n_b denotes the number of blocks in the action interest region, and the HOG feature vector of each block corresponds to a row in $\widehat{\mathbf{X}}_i$. We vectorize HOG features and normalize each vector to unit ℓ_2 norm, forming individual data samples in a matrix $\mathbf{X} \in \mathbb{R}^{m \times N}$, where

$m = n_b \times 9$ and N denotes the total number of frames in the videos, as shown in Fig. 5.1(a).

The MBH descriptor represents the oriented gradients computed separately from the vertical and horizontal components of the optical flow, which is robust to camera and background motion. To extract MBH descriptors, we first split the optical flow field into two scalar fields corresponding to the horizontal and vertical components, which can be regarded as "images" of the motion components. Similar to HOG descriptor extraction, we divide the action interest region of each of the optical flow component image into a grid of blocks, each of size $n_\sigma \times n_\sigma$ pixels. Spatial derivatives are then computed for each block in each optical flow component and orientation information is quantized into 9 bins. Instead of using MBH features of optical flow field between every two video frames separately, we aggregate MBH features of every n_τ adjacent optical flow fields (computed between $n_\tau + 1$ video frames) and the sum is used as the feature of these n_τ optical flow fields. Therefore, the MBH feature of every n_τ adjacent optical flow fields corresponds to a matrix $\widehat{\mathbf{X}}_i \in \mathbb{R}^{n_b \times 18}$, where n_b denotes the number of blocks in the action interest region, and the MBH feature vector of each block corresponds to a row in $\widehat{\mathbf{X}}_i$. We again vectorize the MBH features and normalize each vector to unit ℓ_2 norm, forming MBH feature matrix $\mathbf{X} \in \mathbb{R}^{m \times N}$, where $m = n_b \times 18$ and N again denotes the total number of feature descriptors, see Fig. 5.1(b). Given all the features extracted from the video data, we aim to learn action attributes based on the UoS model, which will be described in the following sections.

5.2 Clustering-Aware Structure-Constrained Low-Rank Representation

In this section, we propose our clustering-aware structure-constrained LRR (CS-LRR) model for learning the action attributes using the feature descriptors extracted from the video data. We begin with a brief review of LRR and SC-LRR since our CS-LRR model extends these models.

5.2.1 Brief Review of LRR and SC-LRR

Consider a collection of N feature vectors in $\mathbb{R}^m$, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, that are drawn from a union of L low-dimensional subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^L$ of dimensions $\{d_\ell\}_{\ell=1}^L$. The task of subspace clustering is to segment the data points according to their respective subspaces. Low-rank representation (LRR) is a recently proposed subspace clustering method [42, 95] that aims to find the lowest-rank representation for the data using a predefined dictionary to reveal the intrinsic geometric structure of the data. Mathematically, LRR can be formulated as the following optimization problem [95]:

$$\widehat{\mathbf{Z}} = \arg\min_{\mathbf{Z}} \operatorname{rank}(\mathbf{Z}) \quad \text{s.t.} \quad \mathbf{X} = \mathbf{A}\mathbf{Z}, \tag{5.1}$$

where $\mathbf{A}$ is a predefined dictionary that linearly spans the data space and $\mathbf{Z}$ is the low-rank representation of the data over $\mathbf{A}$. In practice, the observed data $\mathbf{X}$ are often corrupted by noise. To better handle the noisy data, the LRR problem can be formulated as

$$(\widehat{\mathbf{Z}}, \widehat{\mathbf{E}}) = \arg\min_{\mathbf{Z},\mathbf{E}} \|\mathbf{Z}\|_* + \lambda \|\mathbf{E}\|_\iota \quad \text{s.t.} \quad \mathbf{X} = \mathbf{A}\mathbf{Z} + \mathbf{E}, \tag{5.2}$$

where $\mathbf{E}$ is a matrix representing the approximation errors of the data, the nuclear norm is the convex relaxation of rank operator and $\|\cdot\|_\iota$ indicates a certain regularization strategy involving $\mathbf{E}$. In [95], the $\ell_{2,1}$ norm is used for regularization because of its robustness against corruptions and outliers in data. Finally, λ is a positive parameter that sets the tradeoff between low rankness of $\mathbf{Z}$ and the representation fidelity. In [95], the whole sample set $\mathbf{X}$ is used as the dictionary for clustering, which takes advantage of the self-expressiveness property of data. Once the matrix $\widehat{\mathbf{Z}}$ is available, a symmetric non-negative similarity matrix $\mathbf{W}$ can be defined as $\mathbf{W} = \frac{|\widehat{\mathbf{Z}}| + |\widehat{\mathbf{Z}}^T|}{2}$, where $|\cdot|$ again denotes the element-wise absolute value operation. Finally, spectral clustering [71] can be performed on $\mathbf{W}$ to get the final clustering results.

It has been proved that LRR can achieve a block-diagonal (up to permutation) solution under the condition that the subspaces underlying the data are independent [42, 95]. However, clustering of disjoint subspaces is more desirable in many real situations [13].[1] To improve upon LRR for disjoint subspace clustering, Tang *et al.* [96] proposed structure-constrained LRR (SC-LRR) model, whose learning can be formulated as follows:

$$(\widehat{\mathbf{Z}}, \widehat{\mathbf{E}}) = \underset{\mathbf{Z},\mathbf{E}}{\arg\min} \|\mathbf{Z}\|_* + \alpha\|\mathbf{B} \odot \mathbf{Z}\|_1 + \lambda\|\mathbf{E}\|_{2,1} \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{Z} + \mathbf{E}, \tag{5.3}$$

where α and λ are penalty parameters, $\mathbf{B} \in \mathbb{R}^{N \times N}$ is a predefined weight matrix associated with the data, and $\odot$ denotes the Hadamard product. It has been shown in [96] that by designing some predefined weight matrices, the optimal solution of $\mathbf{Z}$ is block-diagonal for disjoint subspaces when the data are noiseless. In general, the matrix $\mathbf{B}$ imposes restrictions on the solution by penalizing affinities between data samples from different clusters, while rewarding affinities between data samples from the same cluster. The sample set $\mathbf{X}$ is again selected as the dictionary in [96] for clustering.

5.2.2 CS-LRR Model

Almost all the existing subspace clustering methods follow a two-stage approach: (i) learning the coefficient matrix from the data and (ii) applying spectral clustering on the affinity matrix to segment the data. This two-step approach may lead to sub-optimal clustering results because the final clustering result is independent of the optimization problem that is used to obtain the coefficient matrix. We hypothesize that by making the final clustering result dependent on the generation of the optimal coefficient matrix, we will be able to obtain better clustering results. Specifically, suppose we have the coefficient matrix $\widehat{\mathbf{Z}}$ for SC-LRR. Then one can define an affinity matrix $\mathbf{W}$ as $\mathbf{W} = \frac{|\widehat{\mathbf{Z}}|+|\widehat{\mathbf{Z}}^T|}{2}$. We obtain the clustering of the data by applying spectral clustering [71] on $\mathbf{W}$, which solves the following problem:

$$\widehat{\mathbf{F}} = \underset{\mathbf{F}}{\arg\min}\, \mathrm{tr}(\mathbf{F}^T(\mathbf{V} - \mathbf{W})\mathbf{F}) \quad \text{s.t.} \quad \mathbf{F}^T\mathbf{F} = \mathbf{I}, \tag{5.4}$$

[1]Heuristically, a collection of L subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^L$ is said to be independent if the bases of all the subspaces are linearly independent, whereas $\{\mathcal{S}_\ell\}_{\ell=1}^L$ are said to be disjoint if every pair of subspaces are independent; we refer the reader to [13] for formal definitions.

where $\mathbf{F} \in \mathbb{R}^{N\times L}$ is a binary matrix indicating the cluster membership of the data points, i.e., $\forall i$, $f_{i,\ell} = 1$ if $\mathbf{x}_i$ lies in subspace $\mathcal{S}_\ell$ and $f_{i,\ell} = 0$ otherwise. Here, $\mathbf{V} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with its diagonal elements defined as $v_{i,i} = \sum_j w_{i,j}$. The solution of $\mathbf{F}$ consists of the eigenvectors of the Laplacian matrix $\mathbf{M} = \mathbf{V} - \mathbf{W}$ associated with its smallest L eigenvalues. Note that the objective function of (5.4) can also be written as

$$\operatorname{tr}(\mathbf{F}^T(\mathbf{V}-\mathbf{W})\mathbf{F}) = \frac{1}{2}\sum_{i,j} w_{i,j}\|\mathbf{f}^i - \mathbf{f}^j\|_2^2 = \sum_{i,j}|z_{i,j}|(\frac{1}{2}\|\mathbf{f}^i - \mathbf{f}^j\|_2^2) = \|\mathbf{\Theta}\odot\mathbf{Z}\|_1, \quad (5.5)$$

where $\theta_{i,j} = \frac{1}{2}\|\mathbf{f}^i - \mathbf{f}^j\|_2^2$. In order to capture the relation between $\mathbf{Z}$ and $\mathbf{F}$, we imagine that the exact segmentation matrix $\mathbf{F}$ is known. It can be observed that if $\mathbf{x}_i$ and $\mathbf{x}_j$ lie in different subspaces, i.e., $\mathbf{f}^i \neq \mathbf{f}^j$, then we would like to have $z_{i,j} = 0$ for better clustering. Therefore, we can use (5.5) to quantify the disagreement between $\mathbf{Z}$ and $\mathbf{F}$ [98].

The ground truth segmentation matrix $\mathbf{F}$ is of course unknown in practice. In order to penalize the "disagreement" between $\mathbf{Z}$ and $\mathbf{F}$, we propose a *clustering-aware structure-constrained LRR* (CS-LRR) model obtained by solving the following problem:

$$\begin{aligned}&\min_{\mathbf{Z},\mathbf{F},\mathbf{E}} \|\mathbf{Z}\|_* + \alpha\|\mathbf{B}\odot\mathbf{Z}\|_1 + \beta\operatorname{tr}\Big(\mathbf{F}^T(\mathbf{V} - \frac{|\mathbf{Z}| + |\mathbf{Z}^T|}{2})\mathbf{F}\Big) + \lambda\|\mathbf{E}\|_\iota \\ &\text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{Z} + \mathbf{E},\ \mathbf{F}^T\mathbf{F} = \mathbf{I}, \qquad (5.6)\end{aligned}$$

where α, β and λ are penalty parameters. Similar to [96], the (i,j)-th entry of $\mathbf{B}$ is defined as $b_{i,j} = 1 - \exp(-\frac{1-|\mathbf{x}_i^T\mathbf{x}_j|}{\sigma})$, where σ is the mean of all $1-|\mathbf{x}_i^T\mathbf{x}_j|$'s. The CS-LRR model in (5.6) requires knowledge of the number of subspaces L. Initially, we assume to have knowledge of an upper bound on L, which we denote by $L_{\max}$, and we use $L_{\max}$ in (5.6) to learn the representation matrix. In practice, however, one cannot assume knowledge of this parameter a priori. Therefore, we also develop a hierarchical clustering technique to automatically determine $L_{\max}$, which will be discussed in Section 5.3. The CS-LRR model encourages consistency between the representation coefficients and the subspace segmentation by making the similarity matrix more block-diagonal, which can help spectral clustering achieve the best results.

5.2.3 Solving CS-LRR

To solve the optimization problem (5.6), we first introduce an auxiliary variable $\mathbf{Q}$ to make the objective function of (5.6) separable and reformulate (5.6) as

$$\begin{aligned} \min_{\mathbf{Z},\mathbf{F},\mathbf{E}} & \|\mathbf{Z}\|_* + \alpha\|\mathbf{B}\odot\mathbf{Q}\|_1 + \beta\mathrm{tr}\Big(\mathbf{F}^T(\mathbf{V} - \frac{|\mathbf{Q}| + |\mathbf{Q}^T|}{2})\mathbf{F}\Big) + \lambda\|\mathbf{E}\|_\iota \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{X}\mathbf{Z} + \mathbf{E},\ \mathbf{F}^T\mathbf{F} = \mathbf{I},\ \mathbf{Z} = \mathbf{Q}. \end{aligned} \tag{5.7}$$

This problem can be solved by using the linearized alternating direction method (LADM) [99]. Specifically, the augmented Lagrangian function of (5.7) is

$$\begin{aligned} &\mathcal{L}(\mathbf{Z},\mathbf{Q},\mathbf{F},\mathbf{E},\mathbf{\Gamma}_1,\mathbf{\Gamma}_2) \\ &= \|\mathbf{Z}\|_* + \alpha\|\mathbf{B}\odot\mathbf{Q}\|_1 + \beta\mathrm{tr}\Big(\mathbf{F}^T(\mathbf{V} - \frac{|\mathbf{Q}| + |\mathbf{Q}^T|}{2})\mathbf{F}\Big) + \lambda\|\mathbf{E}\|_\iota \\ &\quad + \langle\mathbf{\Gamma}_1, \mathbf{X} - \mathbf{X}\mathbf{Z} - \mathbf{E}\rangle + \langle\mathbf{\Gamma}_2, \mathbf{Z} - \mathbf{Q}\rangle + \frac{\mu}{2}(\|\mathbf{X} - \mathbf{X}\mathbf{Z} - \mathbf{E}\|_F^2 + \|\mathbf{Z} - \mathbf{Q}\|_F^2), \end{aligned} \tag{5.8}$$

where $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$ are matrices of Lagrangian multipliers and μ is a penalty parameter. The optimization of (5.8) can be done iteratively by minimizing $\mathcal{L}$ with respect to $\mathbf{Z}$, $\mathbf{Q}$, $\mathbf{F}$ and $\mathbf{E}$ one at a time, with all other variables being fixed. Note that we also update $\mathbf{V}$ accordingly once we have $\mathbf{Q}$ updated. The constraint $\mathbf{F}^T\mathbf{F} = \mathbf{I}$ in (5.7) is imposed independently in each step of updating $\mathbf{F}$.

Update $\mathbf{Z}$ *while fixing other variables:* When other variables are fixed, the problem of updating $\mathbf{Z}$ in the $(t+1)$-th iteration ($t \geq 0$) is equivalent to minimizing the following function:

$$f(\mathbf{Z}) = \|\mathbf{Z}\|_* + \hbar(\mathbf{Z},\mathbf{Q}^t,\mathbf{E}^t,\mathbf{\Gamma}_1^t,\mathbf{\Gamma}_2^t), \tag{5.9}$$

where $\hbar(\mathbf{Z},\mathbf{Q}^t,\mathbf{E}^t,\mathbf{\Gamma}_1^t,\mathbf{\Gamma}_2^t) = \frac{\mu^t}{2}(\|\mathbf{X} - \mathbf{X}\mathbf{Z} - \mathbf{E}^t + \frac{\mathbf{\Gamma}_1^t}{\mu^t}\|_F^2 + \|\mathbf{Z} - \mathbf{Q}^t + \frac{\mathbf{\Gamma}_2^t}{\mu^t}\|_F^2)$. However, this variant of the problem does not have a closed-form solution. Nonetheless, in the spirit of LADM, $f(\mathbf{Z})$ can also be minimized by solving the following problem:

$$\begin{aligned} \mathbf{Z}^{t+1} &= \arg\min_{\mathbf{Z}} \|\mathbf{Z}\|_* + \langle\nabla_{\mathbf{Z}}\hbar(\mathbf{Z}^t), \mathbf{Z} - \mathbf{Z}^t\rangle + \frac{\eta\mu^t}{2}\|\mathbf{Z} - \mathbf{Z}^t\|_F^2 \\ &= \arg\min_{\mathbf{Z}} \|\mathbf{Z}\|_* + \frac{\eta\mu^t}{2}\|\mathbf{Z} - \mathbf{Z}^t + \frac{\nabla_{\mathbf{Z}}\hbar(\mathbf{Z}^t)}{\eta\mu^t}\|_F^2, \end{aligned} \tag{5.10}$$

where $\nabla_{\mathbf{Z}}\hbar$ is the partial differential of $\hbar$ with respect to $\mathbf{Z}$ and η is a constant satisfying $\eta > \|\mathbf{X}\|^2$. For this problem, $\nabla_{\mathbf{Z}}\hbar = \mu^t\mathbf{X}^T(\mathbf{XZ}-\mathbf{X}+\mathbf{E}^t-\frac{\mathbf{\Gamma}_1^t}{\mu^t})+\mu^t(\mathbf{Z}-\mathbf{Q}^t+\frac{\mathbf{\Gamma}_2^t}{\mu^t})$. Then the closed-form solution for $\mathbf{Z}$ is given as

$$\mathbf{Z}^{t+1} = \Upsilon_{\frac{1}{\eta\mu^t}}(\mathbf{Z}^t - \frac{\nabla_{\mathbf{Z}}\hbar(\mathbf{Z}^t)}{\eta\mu^t}), \tag{5.11}$$

where $\Upsilon(\cdot)$ denotes singular value thresholding operator [100].

Update $\mathbf{Q}$ *while fixing other variables:* When other variables are fixed, the problem of updating $\mathbf{Q}$ is

$$\min_{\mathbf{Q}} \alpha\|\mathbf{B}\odot\mathbf{Q}\|_1 + \beta\mathrm{tr}\Big((\mathbf{F}^t)^T(\mathbf{V}-\frac{|\mathbf{Q}|+|\mathbf{Q}^T|}{2})\mathbf{F}^t\Big) + \frac{\mu^t}{2}\|\mathbf{Q}-(\mathbf{Z}^{t+1}+\frac{\mathbf{\Gamma}_2^t}{\mu^t})\|_F^2. \tag{5.12}$$

According to (5.5), we have $\mathrm{tr}((\mathbf{F}^t)^T(\mathbf{V}-\frac{|\mathbf{Q}|+|\mathbf{Q}^T|}{2})\mathbf{F}^t) = \|\mathbf{\Theta}^t\odot\mathbf{Q}\|_1$, where $\theta_{i,j}^t = \frac{1}{2}\|\mathbf{f}^{t,i}-\mathbf{f}^{t,j}\|_2^2$. Here, $\mathbf{f}^{t,i}$ and $\mathbf{f}^{t,j}$ denote the i-th and j-th row of the matrix $\mathbf{F}^t$, respectively. Therefore, (5.12) can be written as follows:

$$\min_{\mathbf{Q}} \alpha\|(\mathbf{B}+\frac{\beta}{\alpha}\mathbf{\Theta}^t)\odot\mathbf{Q}\|_1 + \frac{\mu^t}{2}\|\mathbf{Q}-(\mathbf{Z}^{t+1}+\frac{\mathbf{\Gamma}_2^t}{\mu^t})\|_F^2, \tag{5.13}$$

which has the following closed-form solution:

$$\mathbf{Q}^{t+1} = \mathcal{T}_{(\frac{\alpha}{\mu^t},\mathbf{B}+\frac{\beta}{\alpha}\mathbf{\Theta}^t)}(\mathbf{Z}^{t+1}+\frac{\mathbf{\Gamma}_2^t}{\mu^t}), \tag{5.14}$$

where the (i,j)-th entry of $\mathcal{T}_{(\tau,\mathbf{H})}(\mathbf{A})$ is given by $T_{\tau h_{i,j}}(a_{i,j})$ with $T_{\tau'}(x) = \max(x-\tau',0)+\min(x+\tau',0)$ [96]. After this, we update $\mathbf{V}^{t+1}$ by setting $\forall i$, $v_{i,i}^{t+1} = \sum_j \frac{|q_{i,j}^{t+1}|+|q_{j,i}^{t+1}|}{2}$.

Update $\mathbf{F}$ *while fixing other variables:* When other variables are fixed, the problem of updating $\mathbf{F}$ is

$$\mathbf{F}^{t+1} = \arg\min_{\mathbf{F}^T\mathbf{F}=\mathbf{I}} \mathrm{tr}(\mathbf{F}^T(\mathbf{V}^{t+1}-\frac{|\mathbf{Q}^{t+1}|+|(\mathbf{Q}^{t+1})^T|}{2})\mathbf{F}). \tag{5.15}$$

Defining $\mathbf{M}^{t+1} = \mathbf{V}^{t+1} - \frac{|\mathbf{Q}^{t+1}|+|(\mathbf{Q}^{t+1})^T|}{2}$, this problem has a closed-form solution that involves eigendecomposition of $\mathbf{M}^{t+1}$. In particular, the columns of $\mathbf{F}^{t+1}$ are given by the eigenvectors of $\mathbf{M}^{t+1}$ associated with its smallest $L_{\max}$ eigenvalues.

Update $\mathbf{E}$ *while fixing other variables:* When other variables are fixed, the problem of updating $\mathbf{E}$ can be written as

$$\mathbf{E}^{t+1} = \arg\min_{\mathbf{E}} \lambda\|\mathbf{E}\|_\iota + \frac{\mu^t}{2}\|\mathbf{E}-\mathbf{C}^{t+1}\|_F^2, \tag{5.16}$$

where $\mathbf{C}^{t+1} = \mathbf{X} - \mathbf{X}\mathbf{Z}^{t+1} + \frac{\mathbf{\Gamma}_1^t}{\mu^t}$. For HOG features, we define $\widehat{\mathbf{E}}_i$ to be the approximation error with respect to $\widehat{\mathbf{X}}_i$ (the matrix version of $\mathbf{x}_i$) and set the error term $\|\mathbf{E}\|_\iota \equiv \sum_{i=1}^N \|\widehat{\mathbf{E}}_i\|_{2,1}$ to ensure robustness against "corruptions" in the orientation of each HOG feature descriptor; this is because the background information is included in the feature vector. Then (5.16) can be decomposed into N independent subproblems. In order to update $\mathbf{e}_i$, we first convert the vector $\mathbf{c}_i^{t+1}$ to a matrix $\widehat{\mathbf{C}}_i^{t+1} \in \mathbb{R}^{n_b \times 9}$ and then solve the following problem:

$$\widehat{\mathbf{E}}_i^{t+1} = \arg\min_{\widehat{\mathbf{E}}_i} \lambda \|\widehat{\mathbf{E}}_i\|_{2,1} + \frac{\mu^t}{2} \|\widehat{\mathbf{E}}_i - \widehat{\mathbf{C}}_i^{t+1}\|_F^2, \tag{5.17}$$

where $\widehat{\mathbf{E}}_i^{t+1} \in \mathbb{R}^{n_b \times 9}$ is the reshaped "image" of the vector $\mathbf{e}_i^{t+1}$. This problem can be solved using [95, Lemma 3.2]. For MBH features, since the noise due to background motion is eliminated, we simply set the error term $\|\mathbf{E}\|_\iota \equiv \|\mathbf{E}\|_{2,1}$; then (5.16) can be written as

$$\mathbf{E}^{t+1} = \arg\min_{\mathbf{E}} \lambda \|\mathbf{E}\|_{2,1} + \frac{\mu^t}{2} \|\mathbf{E} - \mathbf{C}^{t+1}\|_F^2, \tag{5.18}$$

which can also be solved by using [95, Lemma 3.2]. The complete algorithm is outlined in Algorithm 8.

5.3 Hierarchical Subspace Clustering Based on CS-LRR Model

We now introduce a hierarchical subspace clustering algorithm based on CS-LRR approach for learning action attributes at multiple levels of our UoS model and for automatically determining the final number of attributes present in a high-dimensional dataset without prior knowledge. To begin, we introduce some notation used in this section. We define $\boldsymbol{\pi}_{p|\ell}$ to be the set containing the indexes of all $\mathbf{x}_i$'s that are assigned to the ℓ-th subspace at the p-th level ($p \geq 0$) of the hierarchical structure, and let $\mathbf{X}_{p|\ell} = [\mathbf{x}_i : i \in \boldsymbol{\pi}_{p|\ell}] \in \mathbb{R}^{m \times N_{p|\ell}}$ be the corresponding set of signals, where $N_{p|\ell}$ is the number of signals in $\mathbf{X}_{p|\ell}$.[2] Let L_p denote the number of subspaces at the p-th level, then we have $\sum_{\ell=1}^{L_p} N_{p|\ell} = N$ and $\mathbf{X} = \bigcup_{\ell=1}^{L_p} \mathbf{X}_{p|\ell}$ for all p's. The subspace underlying

[2]At level 0, we have only one cluster at the top, i.e., $\boldsymbol{\pi}_{0|1} = \{1, 2, \ldots, N\}$, $\mathbf{X}_{0|1} = \mathbf{X}$ and $N_{0|1} = N$.

Algorithm 8: Solving CS-LRR by LADM

```
Input: The data matrix $\mathbf{X}$ and matrix $\mathbf{B}$, and parameters $L_{\max}$, $\alpha$, $\beta$ and $\lambda$.
Initialize: $\mathbf{Z}^0 = \mathbf{Q}^0 = \mathbf{\Theta}^0 = \mathbf{\Gamma}_1^0 = \mathbf{\Gamma}_2^0 = \mathbf{0}$, $\rho = 1.1$, $\eta > \|\mathbf{X}\|^2$, $\mu^0 = 0.1$,
  $\mu_{\max} = 10^{30}$, $\epsilon = 10^{-7}$, $t = 0$.
1:  while not converged do
2:    Fix other variables and update $\mathbf{Z}$:
      $\mathbf{Z}^{t+1} = \arg\min_{\mathbf{Z}} \|\mathbf{Z}\|_* + \frac{\eta\mu^t}{2}\|\mathbf{Z} - \mathbf{Z}^t + (\mathbf{X}^T(\mathbf{X}\mathbf{Z} - \mathbf{X} + \mathbf{E}^t - \frac{\mathbf{\Gamma}_1^t}{\mu^t}) + (\mathbf{Z} - \mathbf{Q}^t + \frac{\mathbf{\Gamma}_2^t}{\mu^t}))/\eta\|_F^2$.
3:    Fix other variables and update $\mathbf{Q}$:
      $\mathbf{Q}^{t+1} = \arg\min_{\mathbf{Q}} \alpha\|(\mathbf{B} + \frac{\beta}{\alpha}\mathbf{\Theta}^t) \odot \mathbf{Q}\|_1 + \frac{\mu^t}{2}\|\mathbf{Q} - (\mathbf{Z}^{t+1} + \frac{\mathbf{\Gamma}_2^t}{\mu^t})\|_F^2$.
4:    Compute the Laplacian matrix: $\mathbf{M}^{t+1} = \mathbf{V}^{t+1} - \frac{|\mathbf{Q}^{t+1}| + |(\mathbf{Q}^{t+1})^T|}{2}$ and update $\mathbf{F}$ by
      $\mathbf{F}^{t+1} = \arg\min_{\mathbf{F}^T\mathbf{F}=\mathbf{I}} \operatorname{tr}(\mathbf{F}^T\mathbf{M}^{t+1}\mathbf{F})$.
5:    Fix other variables and update $\mathbf{E}$:
      $\mathbf{E}^{t+1} = \arg\min_{\mathbf{E}} \lambda\|\mathbf{E}\|_\iota + \frac{\mu^t}{2}\|\mathbf{E} - (\mathbf{X} - \mathbf{X}\mathbf{Z}^{t+1} + \frac{\mathbf{\Gamma}_1^t}{\mu^t})\|_F^2$.
6:    Update Lagrange multipliers:
      $\mathbf{\Gamma}_1^{t+1} = \mathbf{\Gamma}_1^t + \mu^t(\mathbf{X} - \mathbf{X}\mathbf{Z}^{t+1} - \mathbf{E}^{t+1})$,
      $\mathbf{\Gamma}_2^{t+1} = \mathbf{\Gamma}_2^t + \mu^t(\mathbf{Z}^{t+1} - \mathbf{Q}^{t+1})$.
7:    Update $\mu^{t+1}$ as $\mu^{t+1} = \min(\mu_{\max}, \rho\mu^t)$.
8:    Check convergence conditions and break if
      $\|\mathbf{X} - \mathbf{X}\mathbf{Z}^{t+1} - \mathbf{E}^{t+1}\|_\infty \le \epsilon$, $\|\mathbf{Z}^{t+1} - \mathbf{Q}^{t+1}\|_\infty \le \epsilon$.
9:    Update $t$ by $t = t + 1$.
10: end while
Output: The optimal low-rank representation $\widehat{\mathbf{Z}} = \mathbf{Z}^t$.
```

$\mathbf{X}_{p|\ell}$ is denoted by $\mathcal{S}_{p|\ell}$ and the orthonormal basis of $\mathcal{S}_{p|\ell}$ is denoted by $\mathbf{U}_{p|\ell} \in \mathbb{R}^{m \times d_{p|\ell}}$, where $d_{p|\ell}$ denotes the dimension of the subspace $\mathcal{S}_{p|\ell}$.

We first apply Algorithm 8 to obtain the optimal representation coefficient matrix $\widehat{\mathbf{Z}}$. Then we set the coefficients below a given threshold to zeros, and we denote the final representation matrix by $\widetilde{\mathbf{Z}}$. By defining the affinity matrix $\mathbf{W} = \frac{|\widetilde{\mathbf{Z}}| + |\widetilde{\mathbf{Z}}^T|}{2}$, our hierarchical clustering procedure bears resemblance to the procedure in Chapter 4. At each level $p \ge 0$, we split $\mathbf{X}_{p|\ell}$ into two sub-clusters by applying spectral clustering [71] on $[\mathbf{W}]_{\boldsymbol{\pi}_{p|\ell}, \boldsymbol{\pi}_{p|\ell}}$ (the submatrix of $\mathbf{W}$ whose rows and columns are indexed by $\boldsymbol{\pi}_{p|\ell}$). Note that when $p \ge 2$, we have an additional step which decides whether or not to further divide each single cluster (i.e., subspace) at the p-th level into two clusters (subspaces) at the $(p + 1)$-th level. The cluster $\mathbf{X}_{p|\ell}$ is divisible if and only if (i) the relative representation errors of the data samples using the child subspace are less than the representation errors calculated using the parent subspace by a certain threshold, and (ii) the dimensions of the two child subspaces meet a minimum requirement. Otherwise

Algorithm 9: HCS-LRR (the p-th level, $2 \leq p \leq P-1$)

```
Input: The affinity matrix $\mathbf{W}$ obtained from Algorithm 8. A set of clusters $\{\mathbf{X}_{p|\ell}\}_{\ell=1}^{L_p}$ with their corresponding indexes $\{\boldsymbol{\pi}_{p|\ell}\}_{\ell=1}^{L_p}$, their underlying subspace bases $\{\mathbf{U}_{p|\ell}\}_{\ell=1}^{L_p}$ and $\{g_{p|\ell}\}_{\ell=1}^{L_p}$, and parameters $\gamma$, $\varrho$ and $d_{\min}$.
1:  $\widehat{L} = 0$.
2:  for all $\ell = 1$ to $L_p$ do
3:    if $g_{p|\ell} = 1$ then
4:      $\theta = 0$.
5:      Apply spectral clustering on $[\mathbf{W}]_{\boldsymbol{\pi}_{p|\ell},\boldsymbol{\pi}_{p|\ell}}$ to split $\mathbf{X}_{p|\ell}$ into $\mathbf{X}_{p|\ell} = \boldsymbol{\Sigma}_1 \cup \boldsymbol{\Sigma}_2$ ($\boldsymbol{\pi}_{p|\ell} = \boldsymbol{\chi}_1 \cup \boldsymbol{\chi}_2$).
6:      $\forall c = 1,2$, estimate $d_{\boldsymbol{\Sigma}_c}$ and $\mathbf{U}_{\boldsymbol{\Sigma}_c}$ of $\mathcal{S}_{\boldsymbol{\Sigma}_c}$ using $\boldsymbol{\Sigma}_c$.
7:      $\forall c = 1,2$, compute $\bar{\delta}_c$ and $\bar{\zeta}_c$. If $\frac{\bar{\delta}_c - \bar{\zeta}_c}{\bar{\delta}_c} \geq \varrho$, $\theta = \theta + 1$.
8:      if $\theta \geq 1$ and $\min(d_{\boldsymbol{\Sigma}_1}, d_{\boldsymbol{\Sigma}_2}) \geq d_{\min}$ then
9:        $\forall c = 1,2$, $\mathbf{X}_{p+1|\widehat{L}+c} = \boldsymbol{\Sigma}_c$, $\boldsymbol{\pi}_{p+1|\widehat{L}+c} = \boldsymbol{\chi}_c$, $\mathbf{U}_{p+1|\widehat{L}+c} = \mathbf{U}_{\boldsymbol{\Sigma}_c}$, and $g_{p+1|\widehat{L}+c} = 1$.
10:       $\widehat{L} = \widehat{L} + 2$.
11:     else
12:       $\mathbf{X}_{p+1|\widehat{L}+1} = \mathbf{X}_{p|\ell}$, $\boldsymbol{\pi}_{p+1|\widehat{L}+1} = \boldsymbol{\pi}_{p|\ell}$, $\mathbf{U}_{p+1|\widehat{L}+1} = \mathbf{U}_{p|\ell}$, $g_{p|\ell} = 0$, $g_{p+1|\widehat{L}+1} = 0$, and $\widehat{L} = \widehat{L} + 1$.
13:     end if
14:   else
15:     $\mathbf{X}_{p+1|\widehat{L}+1} = \mathbf{X}_{p|\ell}$, $\boldsymbol{\pi}_{p+1|\widehat{L}+1} = \boldsymbol{\pi}_{p|\ell}$, $\mathbf{U}_{p+1|\widehat{L}+1} = \mathbf{U}_{p|\ell}$, $g_{p+1|\widehat{L}+1} = 0$, and $\widehat{L} = \widehat{L} + 1$.
16:   end if
17: end for
18: $L_{p+1} = \widehat{L}$.
Output: A set of clusters $\{\mathbf{X}_{p+1|\ell}\}_{\ell=1}^{L_{p+1}}$ with their corresponding indexes $\{\boldsymbol{\pi}_{p+1|\ell}\}_{\ell=1}^{L_{p+1}}$, orthonormal bases of the attributes $\{\mathbf{U}_{p+1|\ell}\}_{\ell=1}^{L_{p+1}}$ and $\{g_{p+1|\ell}\}_{\ell=1}^{L_{p+1}}$.
```

the cluster $\mathbf{X}_{p|\ell}$ is a leaf cluster and this cluster will not be divided any further. This process is repeated until we reach a predefined maximum level in the hierarchy denoted by P. The hierarchical subspace clustering algorithm based on CS-LRR model for any level $2 \leq p \leq P-1$ is described in Algorithm 9, which we term HCS-LRR. It is worth noting that the maximum number of leaf clusters is $L_{\max} = 2^P$ in this setting, which we set as a key input parameter for Algorithm 8.

5.4 Attribute Visualization and Semantic Summarization

Given the learned subspaces at different levels of the hierarchical structure, our next goal is to develop a method that helps an expert-in-the-loop to visualize the learned human action attributes, give them semantic labels, and use the labeled attributes to summarize long video sequences of human activities in terms of language at different resolutions. As we have shown previously in Chapter 4, if frame-by-frame silhouette features are used for learning the human action attributes, the attributes (subspaces) can be easily visualized by reshaping the first few vectors of the orthonormal bases of the subspaces into an image format and displaying the scaled versions of these images. However, if other spatial or spatio-temporal features like HOG or MBH are used, the attributes or the subspaces learned using HCS-LRR algorithm cannot be visualized directly by just reshaping each dimension of the subspace in the feature domain.

5.4.1 Visualization of Attributes Using HOG Features

In the case of HOG features, inspired by HOGgles [101], we propose an algorithm to visualize the learned attributes by mapping them back to the pixel domain. In particular, we are interested in building a mapping between the pixel (image) space and the HOG feature space and use this mapping to transform the bases of the HOG feature subspaces into the image space and visualize the attributes. An algorithm based on paired dictionary learning is used to develop this mapping. Concretely, let $\mathbf{X}^I = [\mathbf{x}_1^I, \mathbf{x}_2^I, \ldots, \mathbf{x}_{N_p}^I] \in \mathbb{R}^{m_I \times N_p}$ be the collection of N_p vectorized patches of size $m_I = n_\sigma \times n_\sigma$ pixels from video frames, and $\mathbf{X}^H = [\mathbf{x}_1^H, \mathbf{x}_2^H, \ldots, \mathbf{x}_{N_p}^H] \in \mathbb{R}^{m_H \times N_p}$ be the corresponding HOG feature vectors of the patches in $\mathbf{X}^I$. Here, the dimensionality of the HOG features of each patch m_H depends on the choice of the number of bins in the histogram. For better visualization quality, we extract 18-bin contrast specific HOG features. Hence, $m_H = 18$ in this work. Then, two dictionaries, i.e., overcomplete bases whose columns span the data space, $\mathbf{D}_I \in \mathbb{R}^{m_I \times K}$ and $\mathbf{D}_H \in \mathbb{R}^{m_H \times K}$ consisting of K atoms are learned to represent image space and HOG feature space, respectively, such that the sparse representation of any $\mathbf{x}_i^I \in \mathbf{X}^I$ in terms of $\mathbf{D}_I$ should be the same as that of $\mathbf{x}_i^H \in \mathbf{X}^H$ in terms of $\mathbf{D}_H$. Similar to [102], paired dictionary learning problem can be considered as solving the following optimization problem:

$$\min_{\mathbf{D}_I,\mathbf{D}_H,\{\boldsymbol{\alpha}_i\}_{i=1}^{N_p}} \sum_{i=1}^{N_p} \left(\|\mathbf{x}_i^I - \mathbf{D}_I\boldsymbol{\alpha}_i\|_2^2 + \|\mathbf{x}_i^H - \mathbf{D}_H\boldsymbol{\alpha}_i\|_2^2 \right)$$
$$\text{s.t.} \quad \|\boldsymbol{\alpha}_i\|_1 \le \varepsilon, \forall i, \|\mathbf{d}_{I,k}\|_2 \le 1, \|\mathbf{d}_{H,k}\|_2 \le 1, \forall k, \tag{5.19}$$

where $\boldsymbol{\alpha}_i$ denotes the sparse code with respect to $\mathbf{x}_i^I/\mathbf{x}_i^H$, and $\mathbf{d}_{I,k}$ and $\mathbf{d}_{H,k}$ denote the k-th column of $\mathbf{D}_I$ and $\mathbf{D}_H$, respectively.

Equation (5.19) can be simplified into a standard dictionary learning and sparse coding problem and can be solved using the K-SVD algorithm [16]. Once we have the collection of HOG feature vectors that lie on a subspace, we can find orthogonal basis of the attribute (subspace) using eigenanalysis. However, these orthogonal basis vectors are not guaranteed to be non-negative, which is the characteristic of HOG feature vectors. Therefore, we use Non-negative Matrix Factorization (NMF) [103] to obtain non-negative basis vectors (not necessarily orthogonal) of each subspace in the feature domain. In order to obtain the corresponding pixel values of each of the basis vectors, we split every basis vector into n_b segments and each segment is of length m_H (since the entire basis vector is the concatenation of HOG features extracted from different patches or blocks in a video frame). After this, we infer the sparse representation of every small segment in the basis vector with respect to $\mathbf{D}_H$ using Orthogonal Matching Pursuit (OMP) [104], and use the resulting sparse code on $\mathbf{D}_I$ to recover the pixel values corresponding to that patch (which has m_I values) in the frame. This procedure is repeated for all the segments in the basis vector to obtain its image version for visualization. Finally, the subspace can be labeled by visualizing the subspaces, similar to what was done in the case of silhouette features.

5.4.2 Visualization of Attributes Using MBH Features

Unlike HOG features, it is not feasible to learn the mapping between the feature domain and the pixel domain for MBH features because most of the patches in the pixel domain will be mapped to zero in the MBH feature domain except the blocks in the action interest region. Instead, we store an intermediate output, i.e., the magnitude of the optical flow for visualization purpose. In other words, for every n_τ consecutive optical flow fields, we use the magnitude of the first optical flow field, which can be visualized by an image, as the representative of all these n_τ optical flow fields. Then for every $n_\sigma \times n_\sigma \times n_\tau$ spatio-temporal patch (block) of optical flow fields, from which MBH features are extracted, we have one $n_\sigma \times n_\sigma$ optical flow (magnitude) patch associated with this block. We use MBH features for clustering the data samples into different attributes. Then, we use the optical flow frames corresponding to the data samples in each cluster or attribute to visualize the subspace. The optical flow frames look very similar to the silhouette features with non-zero elements present only in the location where there is movement between the original video frames. Thus, the attributes (subspaces) can be easily visualized by reshaping the first few columns of the orthonormal bases of the subspaces, which are now represented by optical flow frames, into an image format.

5.5 Action Recognition Using Learned Subspaces

In this section, we describe the classification strategy to perform action recognition using the hierarchical union of subspaces learned from Algorithm 9. We assume HCS-LRR ends up with L_P leaf subspaces and the orthonormal bases of these subspaces are being represented by $\{\mathbf{U}_{P|\ell} \in \mathbb{R}^{m \times d_{P|\ell}}\}_{\ell=1}^{L_P}$. We begin our discussion for the classical multi-class closed set recognition problem.

5.5.1 Closed Set Action Recognition Using k-NN

First, we develop a closed set action recognition method based on a k nearest neighbors classifier. Let $\{\mathbf{\Phi}_i \in \mathbb{R}^{m \times \xi_i}\}_{i=1}^{N_T}$ be a collection of N_T labeled training samples, where ξ_i denotes the number of feature vectors of the i-th training video sequence. Given a test video sequence, whose feature vectors are denoted by $\widehat{\mathbf{\Phi}} = [\widehat{\boldsymbol{\phi}}_1, \widehat{\boldsymbol{\phi}}_2, \ldots, \widehat{\boldsymbol{\phi}}_{\widehat{\xi}}]$, we compute the distance between the feature vectors of every training and test sequence as follows. Let $\mathbf{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_\xi]$ be any training sample (we remove subscript i for ease of notation), we first apply Dynamic Time Warping (DTW) [91] to align the two action sequences using the HOG/MBH feature vectors and remove redundant segments at the start and the end of the path. We define $\mathbb{P}_{\mathbf{\Phi},\widehat{\mathbf{\Phi}}} = \{\boldsymbol{\phi}_{a_h}, \widehat{\boldsymbol{\phi}}_{\widehat{a}_h}\}_{h=1}^{H}$ to be the optimal alignment path with length H. Then we assign every vector in these two sequences to the "closest leaf subspace" in $\mathcal{S}_{P|\ell}$'s and we use $\boldsymbol{\psi} \in \mathbb{R}^{\xi}$ and $\widehat{\boldsymbol{\psi}} \in \mathbb{R}^{\widehat{\xi}}$ to denote the vectors which contain the resulting subspace assignment indexes of $\mathbf{\Phi}$ and $\widehat{\mathbf{\Phi}}$, respectively. Based on the optimal alignment path $\mathbb{P}$, the distance $d_F(\mathbf{\Phi}, \widehat{\mathbf{\Phi}})$ between these two action sequences is defined as the average of the normalized distances between the leaf subspaces on the alignment path:

$$d_F(\mathbf{\Phi}, \widehat{\mathbf{\Phi}}) = \frac{\sum_{h=1}^{H} d_u(\mathcal{S}_{P|\psi_{(a_h)}}, \mathcal{S}_{P|\widehat{\psi}_{(\widehat{a}_h)}})}{H}, \tag{5.20}$$

where $d_u(\mathcal{S}_{P|\ell}, \mathcal{S}_{P|\widehat{\ell}}) = \sqrt{1 - \frac{\operatorname{tr}(\mathbf{U}_{P|\ell}^T \mathbf{U}_{P|\widehat{\ell}} \mathbf{U}_{P|\widehat{\ell}}^T \mathbf{U}_{P|\ell})}{\max(d_{P|\ell}, d_{P|\widehat{\ell}})}}$ [44]. Finally, we use the k nearest neighbors (k-NN) classifier to recognize actions based on sequence-to-sequence distances, i.e., a test sequence is declared to be in the class for which the average distance between the test sequence and the k nearest training sequences is the smallest.

5.5.2 Closed Set Action Recognition Using SVM

Next, we describe a closed set action recognition method based on a non-linear support vector machine (SVM) classifier. Given a collection of N_T labeled training samples, denoted by $\{\mathbf{\Phi}_i \in \mathbb{R}^{m \times \xi_i}\}_{i=1}^{N_T}$, we first assign every vector in the training samples to the "closest leaf subspace" in $\mathcal{S}_{P|\ell}$'s, and we use $\{\boldsymbol{\psi}_i \in \mathbb{R}^{\xi_i}\}_{i=1}^{N_T}$ to denote the set of resulting subspace transition vectors. One of the most widely used kernel functions in kernel SVM is Gaussian radial basis function (RBF) kernel $\kappa(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j) = \exp(-\frac{\|\boldsymbol{\psi}_i - \boldsymbol{\psi}_j\|_2^2}{\nu^2})$ [105], where ν is the bandwidth parameter. However, for action recognition based on the UoS model, the subspace transition vectors of different video sequences have different lengths. Hence, we first use DTW [91] on the Grassmann manifold to compute the distance between two action video sequences by only using subspace transition vectors (without aligning the sequences using feature vectors) in Algorithm 7 described in Chapter 4, which is denoted by $d_T(\cdot,\cdot)$, and replace the Euclidean distance in the Gaussian RBF kernel with DTW distance to obtain a Gaussian DTW kernel [106] as $\kappa(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j) = \exp(-\frac{d_T^2(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)}{\nu^2})$. Finally, we use both *one-vs.-all* and *one-vs.-one* approach [105] for classification.

5.5.3 Open Set Action Recognition

We now consider the open set action recognition problem, where we assume to have knowledge of $\mathcal{B}$ known actions in the training stage while new or never-before-seen actions are also encountered in the test stage [107–109]. We first describe our approach for the open set problem based on the k nearest neighbors classifier. Similar to the solution for the traditional multi-class closed set recognition problem described in Section 5.5.1, we first compute distances between every pair of the training video sequences in each class using (5.20). Then for every training sequence, we calculate the average distance between this training sequence and its k nearest neighbors within the same class, and we use φ_s to denote the maximum of all the averaged distances associated with all the training sequences in the s-th class ($s = 1, 2, \ldots, \mathcal{B}$). For a test sequence, we first apply k-NN to assign a "tentative" class membership for this sequence, which is denoted by $\widehat{s}$, and we use $\widehat{\varphi}$ to denote the average distance between this test sequence and its k nearest training sequences in the $\widehat{s}$-th class. Finally, we declare this test sequence to be in the $\widehat{s}$-th class if $\widehat{\varphi} \leq \varphi_{\widehat{s}}\varsigma$, where $\varsigma > 1$ is a predefined thresholding parameter, otherwise it is labeled as a "new" action class.

Next, we discuss our solution for the open set problem based on one-vs.-all SVM. We first train $\mathcal{B}$ one-vs.-all SVM classifiers using training sequences as proposed in Section 5.5.2. Then for a test sequence, we declare it to be in the $\widehat{s}$-th class if (*i*) the classifier associated with the $\widehat{s}$-th action returns the maximum score among all the classifiers and (*ii*) this maximum score is above zero; otherwise, the test sequence is declared to be a "new" action.

5.6 Experimental Results

We assess our proposed method on five human action datasets. Our first objective is to visually interpret and label the learned action attributes from HCS-LRR and to investigate the utility of the multi-resolution action attributes in semantic summarization of long video sequences of complex human activities. The second goal is to evaluate the performance of the learned action attributes in human action recognition/open set

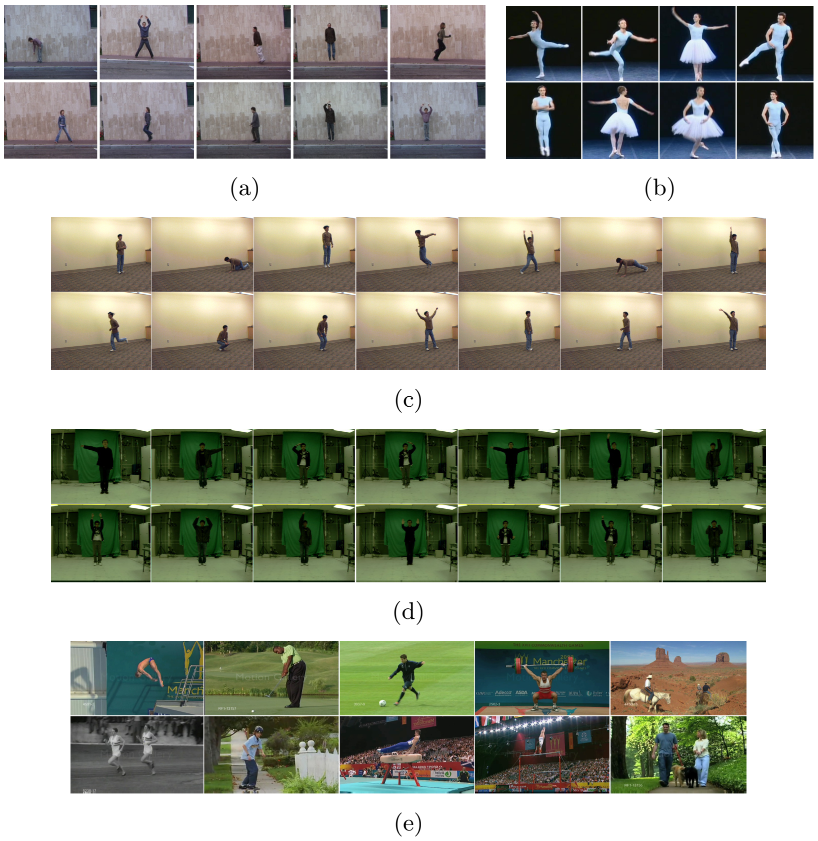


Figure 5.2: Sample frames from the five action recognition datasets used in our experiments: (a) Weizmann dataset; (b) Ballet dataset; (c) UIUC dataset; (d) Keck dataset and (e) UCF Sports dataset.

recognition and to compare our approach with several other union-of-subspaces learning methods.

5.6.1 Datasets

The *Weizmann* dataset [90] consists of 90 low resolution (180×144) video sequences from nine subjects. Each of the subjects performs 10 different actions: walk, run, jump, gallop sideways, bend, one-hand wave, two-hands wave, jump in place, jumping jack, and skip. The camera setting is fixed in this dataset. Sample frames are shown in Fig. 5.2(a).

The *Ballet* dataset [40] contains 44 video sequences of 8 unique actions. The eight actions performed by three subjects are left-to-right hand opening, right-to-left hand opening, standing hand opening, leg swinging, jumping, turning, hopping, and standing still. Fig. 5.2(b) presents the sample frames of each action. Each video may contain several actions in this dataset. In total, there are 59 video clips, each containing only one action.

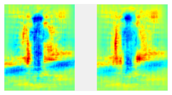
The *UIUC* dataset [110] consists of 532 videos corresponding to eight subjects. In total, there are 14 action classes, including walking, running, jumping, waving, jumping jacks, clapping, jump from situp, raise one hand, stretching out, turning, sitting to standing, crawling, pushing up and standing to sitting (refer to Fig. 5.2(c)). In this work, we use the last section of the dataset, which contains 70 action sequences of all 14 actions performed by only one person and each action is performed five times.

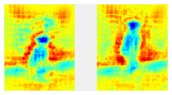
The *Keck* gesture dataset [92] is collected using a camera with 640×480 resolution. It consists of 14 different gestures, including turn left, turn right, attention left, attention right, flap, stop left, stop right, stop both, attention both, start, go back, close distance, speed up, and come near. Example frames of this dataset are shown in Fig. 5.2(d). Each of these 14 actions is performed by three people. In each video sequence, the subject repeats the same gesture three times. Therefore the total number of video sequences with static background is $14 \times 3 \times 3 = 126$.

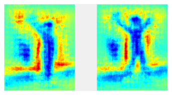
The *UCF Sports* dataset [111] contains 150 video samples, which are collected from various broadcast television channels such as BBC and ESPN. There are 10 action classes included in this dataset: diving, golf swing, kicking, lifting, horse riding, running, skateboarding, swinging-bench, swinging-side and walking, as shown in Fig. 5.2(e). In this work, we exclude the videos belonging to horse riding and skateboarding actions as there is little to no body movement in these two classes. We also exclude two videos in the diving action since there are two subjects in each of those videos. This leaves us with 124 video sequences of 8 actions.

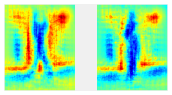
5.6.2 Semantic Labeling and Summarization

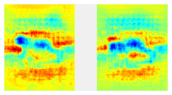
In this section, we study the performance of HCS-LRR using HOG/MBH features for semantic summarization of multiple actions in a long video sequence. We compare the performance of HCS-LRR with several state-of-the-art subspace clustering algorithms such as Low-Rank Representation (LRR) [42], Sparse Subspace Clustering (SSC) [13],

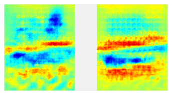

(a) Stand/Clapping

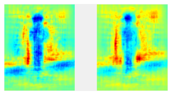

(b) Knee bend

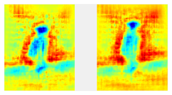

(c) Wave/Jumping jack

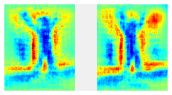

(d) Stretching out/Stand with one hand up

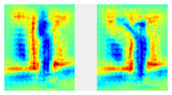

(e) Upper position of push up

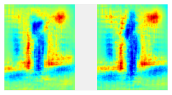

(f) Lower position of push up

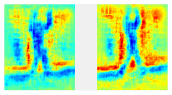

(g) Stand/Clapping

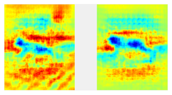

(h) Knee bend

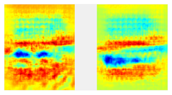

(i) Jumping jack/Stand with two hands out

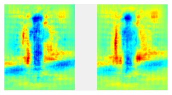

(j) Wave

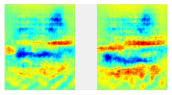

(k) Stand with one hand up

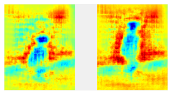

(l) Stretching out/Stand with two hands up

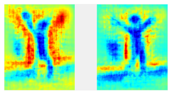

(m) Upper position of push up

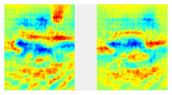

(n) Lower position of push up

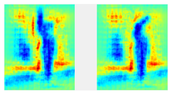

(o) Stand/Clapping

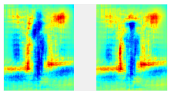

(p) Lower position of push up


(q) Knee bend


(r) Jumping jack


(s) Upper position of push up


(t) Wave


(u) Stand with one hand up/Stand

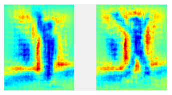

(v) Wave/Jumping jack

Figure 5.3: Visualization and interpretation of attributes learned using the frames from UIUC dataset. (a)-(f) represent the subspaces at the 3rd level of HCS-LRR. (g)-(n) represent the leaf subspaces (4th level) learned by HCS-LRR. (o)-(v) represent the subspaces which are learned using SC-LRR.

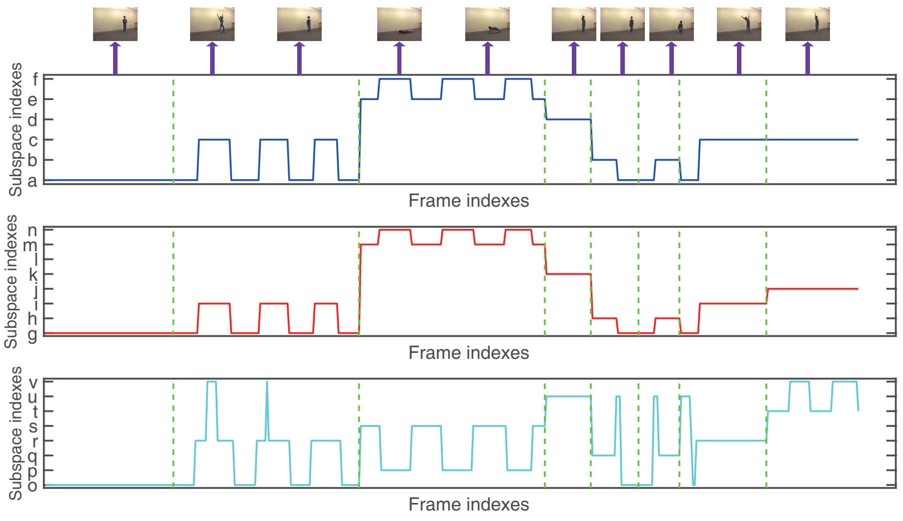
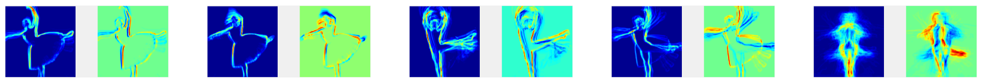
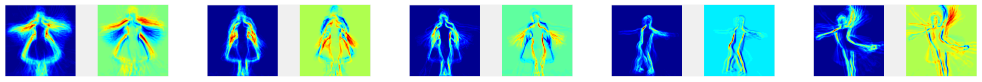


Figure 5.4: Subspace transition of a long sequence using subspaces learned from the 3rd level of HCS-LRR (top), the bottom (4th) level of HCS-LRR (middle), and SC-LRR (bottom). The subspace assignment indexes correspond to the attributes in Fig. 5.3.

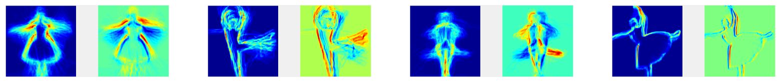
Structure-Constrained LRR (SC-LRR) [96], and Least Square Regression (LSR) [86]. The number of clusters L for these methods is set to be the same number of subspaces generated by HCS-LRR at the bottom-most level. We first focus on UIUC dataset, where we select the first four video sequences of clapping, jumping jacks, pushing up, raise one hand, sitting to standing, standing to sitting, stretching out and waving actions and use the HOG features of these video sequences to learn the attributes. To extract HOG features, we use the bounding boxes provided by the authors of [110] and crop all the sequences into 544×448 aligned frames; then we set $n_\sigma = 32$ and hence $m = 2142$. We apply HCS-LRR with parameters $\alpha = 1.6$, $\beta = 0.5$, $\lambda = 0.5$, $P = 4$, $\gamma = 0.98$, $\varrho = 0.01$ and $d_{\min} = 3$. HCS-LRR returns 6 subspaces at the 3rd level and $L_P = 8$ leaf subspaces at the 4th level. In order to visualize these subspaces, we use patches of size 32×32 and contrast-sensitive HOG features (18-bin histogram) to learn the mapping between the HOG feature domain and the pixel domain, and coupled dictionaries with $K = 1000$ atoms are learned using (5.19). The first two dimensions of the bases of those subspaces in the 3rd and 4th level with their semantic labels are shown in Fig. 5.3 (a)-(f) and Fig. 5.3 (g)-(n), respectively. It can be seen that the attributes (a), (b), (e) and (f) from the 3rd level are leaf nodes and hence, translate as they were to the 4th level. However, the attribute (c) corresponding to Wave/Jumping jack at the 3rd level is further divided into attributes (i) and (j) corresponding to Jumping jack/Stand with two hands out and One-hand wave at the 4th level, respectively. Similarly, the attribute (d) corresponding to Stretching out/Stand with one hand up at the 3rd level is further divided into attributes (k) and (l) corresponding to Stand with one hand up and Stretching out/Stand with two hands up in the bottom level, respectively. To demonstrate semantic summarization using these learned attributes on a complex human activity, we select the last video sequence of each of the aforementioned eight actions as the test sequences and create a long video stream by concatenating these sequences. The semantic summary is illustrated as the subspace transition of the frames in Fig. 5.4, where different actions in the sequence are delimited by green lines. One can summarize the actions in terms of natural language by using the attribute assignment indexes and the corresponding labels. We compare HCS-LRR with other subspace clustering algorithms in terms of action recognition of the test sequences by setting $L = 8$ for other approaches. SC-LRR also gives perfect recognition accuracy based on k-NN classifier. To compare SC-LRR with our technique in terms of semantic summarization, we apply SC-LRR on the training sequences to learn 8 attributes, and the corresponding labeled subspaces are illustrated in Fig. 5.3 (o)-(v). However, compared to the attributes at the bottom level of HCS-LRR, we observe that each of the attributes (u) and (v) learned using SC-LRR can have two totally different interpretations. For instance, attribute (v) is a mixture of "one-hand wave" and "jumping jack" action attributes. There are two advantages of using HCS-LRR model over other techniques. First, the semantic summary of the video sequence can be generated at different resolutions using attributes at different levels of the hierarchical structure. Second, the summary generated using the attributes learned from HCS-LRR is very clear because the actions are well-clustered into different attributes, but it is difficult to generate a summary using the attributes learned from SC-LRR alone due to confusion in the last two attributes. We provide the evidence of these advantages in Fig. 5.4. In short, the semantic interpretation performance using HCS-LRR seems better compared to SC-LRR.



(a) Arabesque (b) Arabesque (c) Développé (d) Arabesque (e) Pirouette with working leg straight



(f) À la seconde (g) Changement (h) Changement (i) Croisé devant (j) Arabesque



(k) À la seconde (l) Développé (m) Pirouette with working leg straight (n) Arabesque

Figure 5.5: Visualization and interpretation of attributes learned using the frames from Ballet dataset. (a)-(g) represent the leaf subspaces learned by HCS-LRR. (h)-(n) represent the subspaces which are learned using SSC.

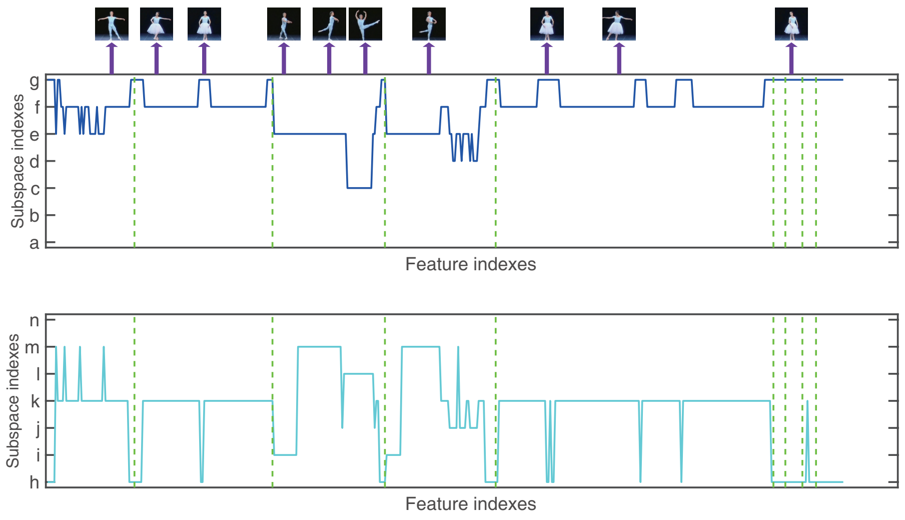


Figure 5.6: Subspace transition of a long sequence using subspaces learned from the 3rd level of HCS-LRR (top) and SSC (bottom). The subspace assignment indexes correspond to the attributes in Fig. 5.5.

As another example, we experiment with Ballet dataset, where we select video clips from standing hand opening, turning, hopping and standing still actions for learning the attributes, with the number of sequences set to be 5, 5, 2 and 6, respectively. We crop all the video frames into 288×288 pixels and extract MBH descriptors with $n_\sigma = 32$ and $n_\tau = 3$; thus $m = 1458$. Then we implement HCS-LRR with parameters $\alpha = 0.8$, $\beta = 0.5$, $\lambda = 0.4$, $P = 3$, $\gamma = 0.98$, $\varrho = 0.05$ and $d_{\min} = 3$. It has $L_P = 7$ leaf subspaces at the 3rd level, and the first two basis vectors of each leaf subspace are illustrated in Fig. 5.5 (a)-(g). Optical flow frames are used for visualization as explained in Section 5.4.2. For testing purpose, we again choose the video clips belonging to the aforementioned actions, with the number of clips set to be 2, 2, 1 and 4, respectively. For these 9 test sequences from these four different actions, attributes learned using SSC with $L = 7$ give perfect (100%) action recognition performance when using k-NN classifier. The subspaces learned using SSC are shown in Fig. 5.5 (h)-(n). We again concatenate all the test video clips to create a long video sequence and visualize the subspace transition of the MBH features of the long sequence in Fig. 5.6. In the case of HCS-LRR, it can be observed that the attribute (b) does not provide any additional information than that is being provided by attribute (a). The attribute (i) learned using SSC is not seen among the attributes at the bottom level of HCS-LRR. In addition, there are no features of the test sequence that get assigned to attributes (a), (b) and (n) due to the significant intraclass variations. In other words, SSC seems to offer better summarization performance than HCS-LRR. However, the caveat here is that in practice, the number of subspaces or attributes is not known a priori for SSC, while HCS-LRR algorithm can automatically give an estimate of L.

5.6.3 Action Recognition

To quantify the discriminative power of the subspaces learned from HCS-LRR, we also perform human action recognition using the action attributes learned by our proposed method. We again compare the performance of HCS-LRR with LRR [42], SSC [13], SC-LRR [96], and LSR [86]. Different from the experiments in Section 5.6.2, the number of clusters L for these algorithms is now set (i) to be the same number of leaf subspaces generated by HCS-LRR (which is denoted by $\langle \textit{Algorithm} \rangle$-$L_P$) and (*ii*) to be the same as the number of actions in the training data (which is denoted by $\langle \textit{Algorithm} \rangle$-$\mathcal{B}$). For all these methods, including HCS-LRR, we first tune the parameters for clustering to achieve their best recognition performance based on k-NN classifier. Once the attributes are learned using these parameters, we further tune the parameters for SVM classifiers to obtain their best recognition performance. We simply fix $\gamma = 0.98$ for all the experiments. It should be noted that "SVM/ova" and "SVM/ovo" in Table 5.1 stand for "one-vs.-all SVM" and "one-vs.-one SVM," respectively.

Closed Set Recognition

In this subsection, we carry out the closed set action recognition experiments on the five datasets described in Section 5.6.1. We begin our experiments on Weizmann dataset, where we use both HOG and MBH features to learn the attributes and evaluate all the subspace/attribute learning approaches based on a leave-one-subject-out experiment. We crop all the sequences into 88×64 aligned frames and HOG features are extracted with blocks of size $n_\sigma = 8$, resulting in $m = 792$. Then we perform HCS-LRR with parameters $\alpha = 1.2$, $\beta = 0.8$, $\lambda = 0.8$, $P = 5$, $\varrho = 0.01$ and $d_{\min} = 5$, in which setting it returns $L_P = 24$ leaf subspaces for final attributes. As shown in Table 5.1, our method significantly outperforms other subspace clustering approaches for all the classifiers. To be specific, the lowest error is achieved by HCS-LRR when used with one-vs.-one SVM classifier and the resulting recognition accuracy is 95.56%. Similarly, we crop all the optical flow fields into 88×64 images and extract MBH features with $n_\sigma = 8$ and $n_\tau = 3$; hence $m = 1584$. We apply HCS-LRR with parameters $\alpha = 0.4$, $\beta = 0.7$, $\lambda = 0.4$, $P = 5$, $\varrho = 0.01$ and $d_{\min} = 12$, obtaining $L_P = 23$ final subspaces. We can see from Table 5.1 that for a fixed classifier, the recognition accuracy of all the approaches is close to each other for 23 clusters. Our algorithm achieves the highest classification accuracy among all the methods for k-NN classifier and one-vs.-all SVM classifier. When using one-vs.-one SVM classifier, LRR-L_P outperforms others and HCS-LRR is only 2.2% (corresponds to 2 samples) behind LRR-L_P. The main reason for k-NN classifier outperforming SVM classifiers is that the original feature vectors are used for aligning the training and test sequences in the case of k-NN classification, and one can get perfect alignment of the video sequences using MBH features on clean datasets such as Weizmann dataset. However, the SVM classifiers only use subspace transition vectors instead of using the original feature vectors for alignment of the sequences via the DTW kernel, which may degrade the recognition performance.

Table 5.1: Closed Set Action Recognition Results (%)

Dataset	Feature	Classifier	Subspace clustering method								
			HCS-LRR	LRR-L_P	LRR-$\mathcal{B}$	SSC-L_P	SSC-$\mathcal{B}$	SC-LRR-L_P	SC-LRR-$\mathcal{B}$	LSR-L_P	LSR-$\mathcal{B}$
Weizmann	HOG	k-NN	**90.00**	73.33	67.78	58.89	52.22	76.67	66.67	67.78	67.78
		SVM/ova	**92.22**	65.56	52.22	58.89	48.89	64.44	63.33	58.89	65.56
		SVM/ovo	**95.56**	82.22	66.67	64.44	51.11	68.89	68.89	71.11	72.22
	MBH	k-NN	**91.11**	88.89	66.67	86.67	65.56	90.00	72.22	90.00	73.33
		SVM/ova	**87.78**	85.56	64.44	85.56	64.44	81.11	68.89	86.67	71.11
		SVM/ovo	85.56	**87.78**	66.67	84.44	61.11	81.11	66.67	85.56	71.11
Ballet	HOG	k-NN	**71.19**	54.24	32.20	52.54	38.98	45.76	49.15	40.68	45.76
		SVM/ova	**67.80**	59.32	30.51	**67.80**	38.98	66.10	42.37	54.24	61.02
		SVM/ovo	62.71	62.71	52.54	**64.41**	42.37	**64.41**	57.63	54.24	61.02
	MBH	k-NN	**69.49**	61.02	54.24	57.63	50.85	57.63	54.24	61.02	52.54
		SVM/ova	**71.19**	69.49	61.02	54.24	20.34	61.02	62.71	67.80	67.80
		SVM/ovo	**69.49**	67.80	61.02	45.76	44.07	67.80	61.02	62.71	67.80
UIUC	HOG	k-NN	**100**	94.29	98.57	98.57	85.71	**100**	95.71	98.57	95.71
		SVM/ova	**100**	77.14	91.43	98.57	81.43	**100**	90.00	**100**	91.43
		SVM/ovo	**100**	92.86	98.57	**100**	78.57	**100**	91.43	**100**	85.71
	MBH	k-NN	**100**	**100**	98.57	**100**	98.57	**100**	82.86	**100**	98.57
		SVM/ova	**100**	**100**	95.71	**100**	**100**	**100**	78.57	**100**	**100**
		SVM/ovo	**100**	**100**	95.71	**100**	**100**	**100**	77.14	**100**	**100**
Keck	MBH	k-NN	**88.10**	83.33	78.57	79.37	83.33	80.16	80.95	76.19	70.63
		SVM/ova	**87.30**	76.98	66.67	74.60	71.43	79.37	80.16	57.14	61.11
		SVM/ovo	**90.48**	76.98	74.60	78.57	73.81	84.92	84.13	70.63	69.05
UCF	MBH	k-NN	**57.26**	50.81	47.58	56.45	47.58	45.97	38.71	55.65	43.55
		SVM/ova	66.13	**67.74**	60.48	66.13	54.84	64.52	50.81	64.52	54.03
		SVM/ovo	**75.81**	75.00	57.26	**75.81**	53.23	73.39	45.16	68.55	50.00

For Ballet dataset, we crop all the video frames into 288×288 pixels and extract HOG descriptor of every frame with $n_\sigma = 32$; hence $m = 729$. Since there is no significant motion between consecutive frames, instead of using HOG features of each frame separately, we take the sum of the HOG features of two adjacent frames at a time and the sum is used as the feature of two adjacent frames. We perform HCS-LRR with parameters $\alpha = 1.5$, $\beta = 0.2$, $\lambda = 2.2$, $P = 4$, $\varrho = 0.05$ and $d_{\min} = 3$. The number of subspaces returned by HCS-LRR at the finest scale is $L_P = 11$. We evaluate the subspace learning approaches using the leave-one-sequence-out scheme. The results presented in Table 5.1 show that the proposed algorithm generates the lowest error rates for k-NN and one-vs.-all SVM classifiers. Both SSC-L_P and SC-LRR-L_P perform the best for one-vs.-one SVM classifier, and our method is only 1 sample lower than these two approaches. After extracting the MBH descriptors with the same parameters as in Section 5.6.2, we set $\alpha = 0.6$, $\beta = 0.2$, $\lambda = 0.2$, $P = 4$, $\varrho = 0.05$ and $d_{\min} = 3$ for HCS-LRR, which gives us $L_P = 9$ leaf subspaces. The results are listed in Table 5.1, from which we make the conclusion that by representing the human actions using the attributes learned by HCS-LRR, we are able to recognize the actions at a superior rate compared to other techniques.

Next, we evaluate our approach on UIUC dataset, where we again use both HOG and MBH features and conduct leave-one-sequence-out experiments. To extract HOG features, we crop all the sequences into 544×448 aligned frames by using the bounding boxes provided by the authors of [110], and follow the same parameter setting as in Section 5.6.2. We perform HCS-LRR with parameters $\alpha = 1$, $\beta = 0.2$, $\lambda = 0.1$, $P = 5$, $\varrho = 0.01$ and $d_{\min} = 3$, in which case it returns $L_P = 18$ leaf subspaces. We crop all the optical flow fields and the action interest region also has 544×448 pixels, then MBH features are extracted with $n_\sigma = 32$ and $n_\tau = 2$; therefore $m = 4284$ in this setting. We set $\alpha = 0.2$, $\beta = 0.1$, $\lambda = 0.1$, $P = 5$, $\varrho = 0.01$ and $d_{\min} = 5$ for HCS-LRR and finally there are $L_P = 26$ subspaces at the bottom-most level. The recognition accuracy for all the methods is reported in Table 5.1. As Table 5.1 shows, both our method and SC-LRR-L_P obtain 100% recognition accuracy for both HOG and MBH features. In general, using the number of clusters that is automatically generated from HCS-LRR will improve the performance for almost all the methods.

To evaluate different methods on Keck dataset, we employ the leave-one-person-out cross validation protocol. We only use MBH features to learn the attributes on this dataset since the background of the aligned sequences is dynamic. We crop all the optical flow fields into 288×288 images and extract MBH features with $n_\sigma = 32$ and $n_\tau = 3$, thereby resulting in $m = 1458$. We implement HCS-LRR with parameters $\alpha = 0.1$, $\beta = 0.1$, $\lambda = 0.2$, $P = 5$, $\varrho = 0.01$ and $d_{\min} = 5$. The number of leaf attributes returned by HCS-LRR is $L_P = 28$. The classification results are summarized in Table 5.1. Our approach shows noticeable improvement in accuracy compared to other state-of-the-art subspace clustering algorithms. The recognition rate reaches 90.48% for HCS-LRR coupled with one-vs.-one SVM classifier, which outperforms SC-LRR-L_P, the second best, by 5.5%.

Finally, we examine the performance of different methods on UCF Sports dataset. We align the sequences by applying the method proposed in [112] and subtract the background using the approach described in [113], then we use the bounding boxes provided by the authors of [111] to crop all the sequences and reshape them into 416×256 frames. Due to the dynamic property of the background, we again only use MBH features to learn the attributes, where the MBH features are extracted with $n_\sigma = 32$ and $n_\tau = 3$; so that $m = 1872$. We investigate the performance of the subspace clustering approaches via the leave-one-sequence-out scheme. We use the following parameters for HCS-LRR: $\alpha = 1.2$, $\beta = 0.1$, $\lambda = 0.3$, $P = 5$, $\varrho = 0.005$ and $d_{\min} = 3$. Finally, the output L_P from HCS-LRR is 26 in this setting. Table 5.1 presents the recognition results for all the methods. For k-NN classifier, our algorithm achieves the highest classification accuracy, with 1 sample higher than the second best, SSC-L_P. The classification performance improves drastically when using SVM classifiers. For one-vs.-one SVM classifier, both HCS-LRR and SSC-L_P achieve 75.81% classification accuracy, but HCS-LRR method has an additional advantage that it can generate the number of clusters automatically without any prior knowledge of the data.

Table 5.2: Open Set Action Recognition Results (%)

Dataset	Classifier	Subspace clustering method				
UIUC	k-NN	HCS-LRR	LRR-L_P	LRR-$\mathcal{B}$	SSC-L_P	SSC-$\mathcal{B}$
		100	90.91	86.36	90.91	86.36
		SC-LRR-L_P	SC-LRR-$\mathcal{B}$	LSR-L_P	LSR-$\mathcal{B}$	
		95.45	95.45	90.91	86.36	
	one-vs.-all SVM	HCS-LRR	LRR-L_P	LRR-$\mathcal{B}$	SSC-L_P	SSC-$\mathcal{B}$
		100	95.45	90.91	90.91	90.91
		SC-LRR-L_P	SC-LRR-$\mathcal{B}$	LSR-L_P	LSR-$\mathcal{B}$	
		100	86.36	95.45	90.91	
Keck	k-NN	HCS-LRR	LRR-L_P	LRR-$\mathcal{B}$	SSC-L_P	SSC-$\mathcal{B}$
		89.58	79.17	77.08	79.17	75.00
		SC-LRR-L_P	SC-LRR-$\mathcal{B}$	LSR-L_P	LSR-$\mathcal{B}$	
		72.92	81.25	77.08	75.00	
	one-vs.-all SVM	HCS-LRR	LRR-L_P	LRR-$\mathcal{B}$	SSC-L_P	SSC-$\mathcal{B}$
		89.58	83.33	85.42	79.17	77.08
		SC-LRR-L_P	SC-LRR-$\mathcal{B}$	LSR-L_P	LSR-$\mathcal{B}$	
		85.42	87.50	81.25	81.25	

Open Set Recognition

One important advantage of learning human action attributes instead of representing an entire action as a single feature vector is that an action that is absent in the training stage can be identified as a new action class if it can be represented by the attributes learned from other actions in the training data. Here, we show the effectiveness of HCS-LRR for the open set action recognition problem. Consider a collection of N_k test samples from $\mathcal{B}$ known classes and N_u test samples from $\mathcal{U}$ unknown classes, which are denoted by $\{(\mathbf{\Phi}_i, \vartheta_i)\}_{i=1}^{N_k+N_u}$, where $\vartheta_i \in \{1, 2, \dots, \mathcal{B}\}$ for $i \leq N_k$ and $\vartheta_i = \textit{new}$ for $i > N_k$. We can measure the performance of our algorithm under the classification rule $\mathcal{C}$ as $\epsilon_o = \frac{\sum_{i=1}^{N_k+N_u}[\mathcal{C}(\mathbf{\Phi}_i)=\vartheta_i]}{N_k+N_u}$.

UIUC dataset is used for the first set of experiments. From the 70 sequences in the dataset, we select the first four video sequences of all classes except "walking" and "standing to sitting" actions for training and the remaining 22 sequences for testing; thus $\mathcal{B} = 12$, $\mathcal{U} = 2$, $N_k = 12$ and $N_u = 10$. We extract HOG features of the training sequences as in Section 5.6.3 and perform HCS-LRR with parameters $\alpha = 0.8$, $\beta = 0.1$, $\lambda = 0.4$, $P = 5$, $\varrho = 0.01$ and $d_{\min} = 3$, which in turn provides us with $L_P = 19$ leaf subspaces. We can observe from Table 5.2 that HCS-LRR achieves perfect accuracy for both classifiers. Moreover, the recognition accuracy is improved when we set $L = 19$ instead of 12 for other subspace clustering approaches.

The second set of experiments are performed on Keck dataset. We select the video sequences associated with the last two subjects from all classes except "come near" action for training and the remaining 48 sequences for testing; hence $\mathcal{B} = 13$, $\mathcal{U} = 1$, $N_k = 39$ and $N_u = 9$. We extract MBH features of the training sequences as in Section 5.6.3 and set $\alpha = 0.1$, $\beta = 0.9$, $\lambda = 0.8$, $P = 5$, $\varrho = 0.01$ and $d_{\min} = 5$ for HCS-LRR. There are $L_P = 18$ attributes learned from HCS-LRR at the bottom level. Table 5.2 summarizes the ϵ_o's of all the algorithms for both classifiers. As can be seen, HCS-LRR outperforms other subspace clustering algorithms, which again proves the effectiveness of our method. In addition, the performance of other subspace clustering algorithms can be significantly enhanced when we use one-vs.-all SVM for recognition.

Chapter 6
Conclusion and Future Work

In this thesis, we considered the problem of learning the geometric structure of high-dimensional data. We proposed novel geometric models and algorithms based on the union-of-subspaces model to learn the geometry efficiently.

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

In the second part of the thesis, we put forth different models for learning the human action attributes from video data based on UoS model. We first proposed Hierarchical Sparse Subspace Clustering (HSSC) to learn the attributes, which does not need the number of clusters to be specified and does not require labeled training data to learn human action attributes, while avoiding generation of trivial attributes. Then we proposed an extension of the canonical low-rank representation (LRR), termed the clustering-aware structure-constrained low-rank representation (CS-LRR) model, for unsupervised data-driven human action attribute learning. The proposed CS-LRR model incorporates spectral clustering into the learning framework, which helps spectral clustering achieve the best clustering results. Moreover, we introduced a hierarchical subspace clustering approach based on CS-LRR model, called HCS-LRR model, which again does not need the number of clusters to be specified a priori. Experimental results on real video datasets showed the effectiveness of our clustering approach and its superiority over the state-of-the-art union-of-subspaces learning algorithms for its utility in semantic summarization of complex human activities at multiple resolutions as well as human action recognition. One of the interesting avenues of future work involves the development of a robust, spatio-temporal hierarchical tensor-UoS framework for learning of semantically interpretable human action attributes and recognition of human activities from multimodal video data.

References

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

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

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

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

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

[6] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, “Kernel PCA and de-noising in feature spaces,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 1998, pp. 536–542.

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

[8] M. A. Turk and A. P. Pentland, “Face recognition using eigenfaces,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 1991, pp. 586–591.

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

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

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

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

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

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

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

[16] 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.

[17] 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.

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

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

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

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

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

[29] 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.

[30] 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.

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

[32] 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.

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

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

[35] 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.

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

[37] 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.

[38] Y.-G. Jiang, S. Bhattacharya, S.-F. Chang, and M. Shah, "High-level event recognition in unconstrained videos," *Int. J. Multimed. Inf. Retr.*, vol. 2, no. 2, pp. 73–101, 2013.

[39] J. Liu, B. Kuipers, and S. Savarese, "Recognizing human actions by attributes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2011, pp. 3337–3344.

[40] A. Fathi and G. Mori, "Action recognition by learning mid-level motion features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2008, pp. 1–8.

[41] J. Liu, Y. Yang, and M. Shah, "Learning semantic visual vocabularies using diffusion distance," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2009, pp. 461–468.

[42] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, 2013.

[43] 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.

[44] 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.

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

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

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

[48] K. Pelckmans, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Convex clustering shrinkage," in *Proc. Statist. Optim. Cluster. Workshop (PASCAL)*, 2005.

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

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

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

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

[53] 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. Comput. Vis. Pattern Recognit. (CVPR)*, 2003, pp. 11–18.

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

[55] E. Levina and P. J. Bickel, "Maximum likelihood estimation of intrinsic dimension," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2004, pp. 777–784.

[56] 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.

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

[58] 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.

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

[60] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 2, pp. 303–353, 1998.

[61] A. Nedić and D. P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM J. Optim.*, vol. 12, no. 1, pp. 109–138, 2001.

[62] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Proc. Allerton Conf. Communication, Control, and Computing*, 2010, pp. 704–711.

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

[64] 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.

[65] B. Eriksson, L. Balzano, and R. Nowak, "High-rank matrix completion," in *Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2012, pp. 373–381.

[66] C. McDiarmid, "On the method of bounded differences," *Surveys in Combinatorics*, vol. 141, pp. 148–188, 1989.

[67] C. J. C. Burges, "Simplified support vector decision rules," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 1996, pp. 71–77.

[68] Y. Rathi, S. Dambreville, and A. Tannenbaum, "Statistical shape analysis using kernel PCA," in *Proc. SPIE*, vol. 6064, 2006.

[69] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, 1994.

[70] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[71] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2001, pp. 849–856.

[72] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2004, pp. 41–48.

[73] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.

[74] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2009, pp. 396–404.

[75] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "ℓ_p-norm multiple kernel learning," *J. Mach. Learn. Res.*, vol. 12, pp. 953–997, 2011.

[76] W. Niu, J. Long, D. Han, and Y.-F. Wang, "Human activity detection and recognition for video surveillance," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, 2004, pp. 719–722.

[77] J. K. Aggarwal, M. S. Ryoo, and K. Kitani, "Frontiers of human activity analysis," *Tutorials of Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011.

[78] M. S. Ryoo and L. Matthies, "First-person activity recognition: What are they doing to me?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2013, pp. 2730–2737.

[79] Y. Zhu, N. M. Nayak, and A. K. Roy-Chowdhury, "Context-aware modeling and recognition of activities in video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2013, pp. 2491–2498.

[80] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A survey on visual content-based video indexing and retrieval," *IEEE Trans. Syst., Man, Cybern., Syst Part C*, vol. 41, no. 6, pp. 797–819, 2011.

[81] U. Akdemir, P. Turaga, and R. Chellappa, "An ontology based approach for activity recognition from video," in *Proc. 16th ACM Int. Conf. Multimedia*, 2008, pp. 709–712.

[82] N. D. Rodríguez, M. P. Cuéllar, J. Lilius, and M. D. Calvo-Flores, "A survey on ontologies for human behavior recognition," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 43:1–43:33, 2014.

[83] D. Ramanan and D. A. Forsyth, "Automatic annotation of everyday movements," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2003, pp. 1547–1554.

[84] J. Liu and M. Shah, "Learning human actions via information maximization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2008, pp. 1–8.

[85] J. Yuan, M. Yang, and Y. Wu, "Mining discriminative co-occurrence patterns for visual recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2011, pp. 2777–2784.

[86] C. Lu, H. Min, Z. Zhao, L. Zhu, D. Huang, and S. Yan, "Robust and efficient subspace segmentation via least squares regression," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2012, pp. 347–360.

[87] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2005, pp. 886–893.

[88] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2006, pp. 428–441.

[89] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[90] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 12, pp. 2247–2253, 2007.

[91] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 26, no. 1, pp. 43–49, 1978.

[92] Z. Lin, Z. Jiang, and L. S. Davis, “Recognizing actions by shape-motion prototype trees,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2009, pp. 444–451.

[93] C.-C. Chen, M. S. Ryoo, and J. K. Aggarwal, “UT-Tower dataset: Aerial view activity classification challenge,” 2010, http://cvrc.ece.utexas.edu/SDHA2010/Aerial_View_Activity.html.

[94] L. Zhuang, S. Gao, J. Tang, J. Wang, Z. Lin, Y. Ma, and N. Yu, “Constructing a nonnegative low-rank and sparse graph with data-adaptive features,” *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3717–3728, 2015.

[95] G. Liu, Z. Lin, and Y. Yu, “Robust subspace segmentation by low-rank representation,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 663–670.

[96] K. Tang, R. Liu, Z. Su, and J. Zhang, “Structure-constrained low-rank representation,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2167–2179, 2014.

[97] H. Gao, F. Nie, X. Li, and H. Huang, “Multi-view subspace clustering,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 4238–4246.

[98] C.-G. Li and R. Vidal, “Structured sparse subspace clustering: A unified optimization framework,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 277–286.

[99] Z. Lin, R. Liu, and Z. Su, “Linearized alternating direction method with adaptive penalty for low-rank representation,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2011, pp. 612–620.

[100] J.-F. Cai, E. J. Candès, and Z. Shen, “A singular value thresholding algorithm for matrix completion,” *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, 2010.

[101] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba, “HOGgles: Visualizing object detection features,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2013, pp. 1–8.

[102] J. Yang, J. Wright, T. S. Huang, and Y. Ma, “Image super-resolution via sparse representation,” *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, 2010.

[103] D. D. Lee and H. S. Seung, “Learning the parts of objects by nonnegative matrix factorization,” *Nature*, vol. 401, pp. 788–791, 1999.

[104] J. A. Tropp and A. C. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[105] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[106] S. Gudmundsson, T. P. Runarsson, and S. Sigurdsson, “Support vector machines and dynamic time warping for time series,” in *Proc. IEEE Int. Joint Conf. Neu. Net. (IJCNN)*, 2008, pp. 2772–2776.

[107] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 7, pp. 1757–1772, 2013.

[108] L. P. Jain, W. J. Scheirer, and T. E. Boult, "Multi-class open set recognition using probability of inclusion," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 393–409.

[109] A. Bendale and T. Boult, "Towards open world recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1893–1902.

[110] D. Tran and A. Sorokin, "Human activity recognition with metric learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2008, pp. 548–561.

[111] M. Rodriguez, J. Ahmed, and M. Shah, "Action MACH: A spatio-temporal maximum average correlation height filter for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2008, pp. 1–8.

[112] X. Zhou, C. Yang, and W. Yu, "Moving object detection by detecting contiguous outliers in the low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 3, pp. 597–610, 2013.

[113] X. Ye, J. Yang, X. Sun, K. Li, C. Hou, and Y. Wang, "Foreground-background separation from video clips via motion-assisted matrix restoration," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 11, pp. 1721–1734, 2015.