# Beyond Linear Subspace Clustering: A Comparative Study of Nonlinear Manifold Clustering Algorithms

Maryam Abdolali[*] Nicolas Gillis[*]
Department of Mathematics and Operational Research
Faculté Polytechnique, Université de Mons
Rue de Houdain 9, 7000 Mons, Belgium

**Abstract**

Subspace clustering is an important unsupervised clustering approach. It is based on the assumption that the high-dimensional data points are approximately distributed around several low-dimensional linear subspaces. The majority of the prominent subspace clustering algorithms rely on the representation of the data points as linear combinations of other data points, which is known as a self-expressive representation. To overcome the restrictive linearity assumption, numerous nonlinear approaches were proposed to extend successful subspace clustering approaches to data on a union of nonlinear manifolds. In this comparative study, we provide a comprehensive overview of nonlinear subspace clustering approaches proposed in the last decade. We introduce a new taxonomy to classify the state-of-the-art approaches into three categories, namely locality preserving, kernel based, and neural network based. The major representative algorithms within each category are extensively compared on carefully designed synthetic and real-world data sets. The detailed analysis of these approaches unfolds potential research directions and unsolved challenges in this field.

**Keywords:** subspace clustering, nonlinear subspace clustering, manifold clustering, Laplacain regularization, Kernel learning, unsupervised deep learning, neural networks

## 1 Introduction

Understanding and processing high-dimensional data is a key component of numerous applications in many domains including machine learning, signal processing and computer vision. However, analyzing high-dimensional data using classical data mining algorithms is not only challenging due to computational costs but it can also easily lead to the well-known problem of the *curse of dimensionality* [8]. Luckily, in most of applications, the data often have fewer degrees of freedom than the ambient dimension and they can be approximately represented by a small number of features; see [106, 105]. This led to the development of a vast variety of algorithms for the long-standing problem of extracting latent structures from high-dimensional data; see for example [109] and references therein.

The main focus of the majority of these algorithms is to fit a *single* low-dimensional linear subspace to the data, with principal component analysis (PCA) being the most well-known pioneer algorithm in

[*]Emails: {maryam.abdolali, nicolas.gillis}@umons.ac.be. The authors acknowledge the support by the European Research Council (ERC starting grant no 679515), and by the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO) under EOS Project no O005318F-RG47.

this area [46]. However, the data often belongs to multiple categories with different intrinsic structures, and modeling the data using only *one* low-dimensional subspace might be too restrictive [108]. In fact, in many applications, the data is better represented by *multiple* subspaces. Representing the data using a union of multiple subspaces gave rise to *linear subspace clustering* [15, 9]; see [107] for a survey paper.

**Definition 1** (Linear subspace clustering (linear SC))**.** *Let $X \in \mathbb{R}^{d \times n}$ be the input high-dimensional data, with $d$-dimensional data points as its columns. Suppose the data points are distributed around $c$ unknown linear subspaces $S_1, S_2, \dots, S_c$ with intrinsic dimensions $d_1, d_2, \dots, d_c$, respectively, such that $d_i \ll d$ for all $i \in \{1, \dots, c\}$. The problem of subspace clustering is defined as segmenting the data points based on their corresponding subspaces and estimating the parameters of each subspace.*

In the literature, linear SC is usually referred to as SC but, to avoid any confusion, we refer to it as linear SC in this survey. Linear SC is a clustering framework where the data points are grouped together based on the underlying subspaces they belong to. In other words, the similarity between the data points is measured by how well they fit within a low-dimensional linear subspace. Modeling data with multiple subspaces has many applications in image and signal processing. In fact, data points are often collected from multiple classes/categories, and extracting latent low-dimensional structures within each class/category results in more meaningful and more compressed representations. For example, given a collection of facial images taken with variations in the lightening, cluserting these images according to the person they represent can be modeled as a linear SC problem. In fact, under the Lambertian surface assumption, facial images of a single individual under different illumination conditions lie on a roughly nine-dimensional subspace which has far less effective dimensions than the original ambient space (usually an image consists of thousands of pixels) [4]. Hence, facial images of various people under different lightening conditions can be approximated by multiple low-dimensional subspaces. Other notable examples of data that can be approximated by multiple low-dimensional subspaces include segmenting trajectories of moving objects in a video [100], clustering images of hand-written digits [32], and partitioning sequences of video frames into semantic parts [109].

Driven by this wide range of applications, many efficient algorithms for linear SC have been developed. These algorithms are divided into four categories [107, 21]: (i) iterative, (ii) statistical, (iii) algebraic, and (iv) spectral clustering based approaches. Let us briefly describe these four categories:

(i) Iterative approaches were among the first ones for linear SC [9, 36, 104]. Inspired by iterative refinement in centroid-based clustering, such as k-means, two steps are carried out alternatively: assigning each data point to the closest subspace, and updating the parameters of each subspace based on the points assigned to it (e.g., using the truncated singular value decomposition). However, iterative approaches are sensitive to initialization and parameters such as the dimensions and the number of subspaces (these are usually not available in most applications).

(ii) Statistical approaches [99, 28], which suffer from the same drawbacks as iterative ones, usually assume that the distribution of the data within each subspace is Gaussian, and hence the linear SC problem is mapped into the iterative problem of fitting a mixture of Gaussian distribution to the data using expectation maximization.

(iii) Some approaches use algebraic matrix factorization to perform the data segmentation [16, 47]. These approaches are not only sensitive to noise but are also based on the assumption that the underlying subspaces are independent. In related algebraic-geometric approaches [108, 103], the association of the data points to the subspaces are revealed by fitting a polynomial to the data. These approaches are computationally expensive and sensitive to noise.

(iv) Spectral clustering based algorithms are the most popular and successful linear SC approaches. They have received significant attention over the last decade. These approaches are based on recent advances in sparse and low-rank representation [39, 118, 146]. They first learn a directed weighted graph representing the interaction between the data points, and then apply spectral clustering to segment this graph; see Section 2 for more details.

Even though linear SC approaches have achieved impressive performances in several applications, the global *linearity* assumption is somewhat strong. Assuming that the data points lie close to multiple *linear* subspaces is limiting and might be violated in some applications. In order to extend the applicability of linear SC in real-world problems, there has been numerous attempts to generalize linear SC for data lying on the union of nonlinear subspaces or manifolds. This is the main topic of this survey paper.

**Our contributions** The main contributions of this paper are summarized as follows:

- We propose a new taxonomy to classify nonlinear SC algorithms, divided them into three categories: (1) locality preserving, (2) kernel based, and (3) neural network based
- We provide a comprehensive overview of these nonlinear SC algorithms.
- We numerically analyze and compare the representative algorithms in each category. This survey can be considered as a guiding tutorial for understanding and developing nonlinear SC algorithms.

**Outline of the survey** The rest of the paper is organized as follows. Since spectral clustering based linear SC algorithms are at the core of the majority of existing nonlinear SC methods, we briefly introduce and review the most important ones in Section 2. Section 3 provides a detailed overview of nonlinear SC methods, based on our new taxonomy that divides them into three categories. Section 4 discusses the computational cost of these methods, while Section 5 provides a numerical comparison of the main representative methods on synthetic and real-world data sets. In Section 6, we discuss the existing challenges in dealing with nonlinear structures in data, and future research directions. We conclude the paper in Section 7.

## 2 Linear subspace clustering based on spectral clustering

Among the wide variety of linear SC approaches, the current state-of-the-art algorithms are based on spectral clustering. As illustrated in Figure 1, these algorithms follow a two-step strategy:

- In the first step, an affinity matrix is constructed: each vertex in the corresponding graph corresponds to a data point while edges connect similar data points.
- In the second step, spectral clustering is applied on the affinity matrix to segment the data points.

In the next two sections, we describe these two steps in more details.

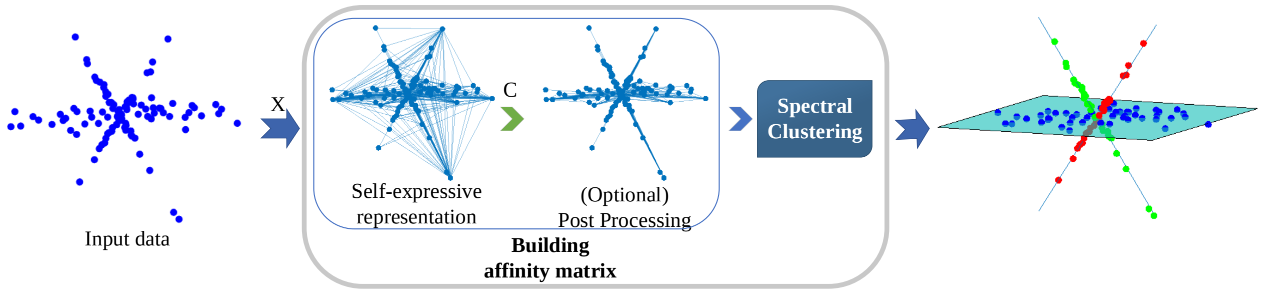


Figure 1: The overall procedure of linear SC approaches based on spectral clustering. In the first step of these approaches, an affinity graph representing the input data is constructed, which is sometimes post-processed to improve the quality of the representation (e.g., using sparsification). In the second step, a spectral clustering algorithm is applied to obtain the clusters.

## 2.1 Building the affinity matrix

The first and undoubtedly main step of linear SC algorithms based on spectral clustering is to build an affinity matrix. The affinity matrix should reveal the *pairwise* similarities between the data points, that is, the data points from the same cluster/subspace should be highly connected with high similarities, as opposed to points from different subspaces. The construction of this affinity matrix is typically carried out using a self-expressive representation; this is discussed in Section 2.1.1. It is sometimes followed by a post-processing step to polish the obtained pairwise similarities; this is discussed in Section 2.1.2.

**Remark 1.** *Building the affinity matrix is not only useful for clustering. It can be used in other contexts where it is useful to model the input data as a graph by capturing pairwise relationships between the data points [14]. In other words, this first step of spectral based SC approaches is also used in other applications that model and analyze the structure within the data [87].*

### 2.1.1 Self-expressive representation

Constructing a pairwise affinity matrix that reflects the multiple subspace structure of the data points is the main challenge in the linear SC algorithms based on spectral clustering. The affinity matrix is often constructed by exploiting the *self-expressiveness* property of the data points. This property, which is known as *collaborative representation* in the sparse representation literature [141], is based on the fact that each data point can be represented as a linear combination of other data points in the same subspace. Mathematically, for all $i \in \{1, \dots, n\}$,

$$X(:,i) = XC(:,i) \quad \text{with} \quad C(i,i) = 0, \tag{1}$$

where $C \in \mathbb{R}^{n \times n}$ is the coefficient matrix, $C(i,i)$ is the $i$-th diagonal entry of $C$, and $X(:,i)$ is the $i$-th column of $X$. The condition $C(i,i) = 0$ for all $i \in \{1, \dots, n\}$ eliminates the trivial solution of expressing each point by itself. However, there are typically infinitely many solutions to (1) and many of them might not be *subspace preserving*, that is, all data point might not be expressed using a linear combination of data points belonging only to the *same* subspace. Let us formally define this concept.

**Definition 2** (Subspace preserving representation)**.** *Given the data matrix $X \in \mathbb{R}^{d\times n}$ drawn from the union of $c$ subspaces $S_1, S_2, \dots, S_c$, let $C \in \mathbb{R}^{n\times n}$ be the coefficient matrix corresponding to the output of a linear SC algorithm based on self-expressiveness; see* (1)*. The matrix $C$ is subspace preserving if for any data point $X(:, i) \in S_\ell$, $C(i, j) = 0$ for all $j$ such that $X(:, j) \notin S_\ell$. In other words, for all $i \in \{1, \dots, n\}$, the nonzero elements in $C(:, i)$ only correspond to data points from the same subspace as $X(:, i)$.*

To fulfill the goal of linear SC and reveal the association of each data point to the underlying subspace, the coefficient matrix should be (approximately) subspace preserving. Hence, to ensure that the obtained coefficient matrix is subspace preserving, several regularizations on the coefficient matrix are used in the literature, including sparsity [21] and low-rankness [59]. In general, the linear SC problem based on self-expressiveness is formulated as follows:

$$\min_{C\in\mathbb{R}^{n\times n}, E\in\mathbb{R}^{d\times n}} f(C) + \lambda g(E) \quad \text{such that} \quad X = XC + E \text{ and } \operatorname{diag}(C) = 0, \tag{2}$$

where $f$ is a regularization function for the coefficient matrix (see below), $E \in \mathbb{R}^{d\times n}$ models the noise, $g$ is a regularization function for modeling the noise (for example the $\ell_1$ norm for sparse gross noise, or the Frobenius norm for Gaussian noise), and $\operatorname{diag}(C)$ is the vector containing the diagonal entries of $C$. The conditions under which the obtained coefficient matrix is subspace preserving strongly depends on the regularization function $f$. Table 1 summarizes the most common regularization functions and the theoretical conditions for the corresponding coefficient matrix to be subspace preserving. In this table $||\cdot||_0$, $||\cdot||_1$, $||\cdot||_*$ and $||\cdot||_F^2$ indicate the $\ell_0$ norm (the number of nonzero entries), the component-wise $\ell_1$ norm (the sum of the absolute value of the matrix entries), the nuclear norm (the sum of singular values), and the squared Frobenius norm (the sum of the squares of the matrix entries), respectively.

**Effect of regularization on subspace preserving representations** The difficulty of linear SC depends on several factors, these include the arrangement of the subspaces, the separation between subspaces, the distribution of the data points within each subspace, the number of points per subspace, and the noise level. There are three main arrangements of subspaces which play a key role in identifying the subspace recovery conditions: independent, disjoint, and intersecting (or overlapping) subspaces. These arrangements are defined as follows:

**Definition 3** (Independent subspaces)**.** *A collection of $c$ subspaces $S_1, \dots, S_c$ are said to be independent if $\dim(\oplus_{i=1}^c S_i) = \sum_{i=1}^c \dim(S_i)$, where $\oplus$ denotes the direct sum between subspaces, and $\dim(S)$ is the dimension of $S$.*

**Definition 4** (Disjoint subspaces)**.** *A collection of $c$ subspaces are disjoint if $\dim(S_i \oplus S_j) = \dim(S_i) + \dim(S_j)$ and $S_i \cap S_j = \{0\}$ for all $i \neq j$.*

**Definition 5** (Intersecting subspaces)**.** *A collection of $c$ subspaces are intersecting/overlapping if $1 \leq \dim(S_i \cap S_j) < \min\{\dim(S_i), \dim(S_j)\}$ for some $i \neq j$.*

Independent subspaces (orthogonal subspaces being a special case) are the easiest to separate, and hence most regularizations are guaranteed to be subspace preserving in this case, at least in noiseless conditions; see Table 1. An example is two distinct lines in a plane.

Clustering data from disjoint subspaces is a more challenging scenario. An example are three distinct lines in a plane. Sparsity regularization based on the $\ell_0$ and $\ell_1$ norms is the only one proven

Table 1: Major linear SC models based on spectral clustering. They differ in the regularization function $f$ used for the coefficient matrix $C$ in (2), and guarantee to provide a subspace preserving representation under different conditions.

| **Method** | **Regularization Function $f$** | **Available subspace preserving guarantees** |
|---|---|---|
| Sparse Subspace Clustering (SSC) [21] | $\|\|C\|\|_1$ | independent, disjoint and intersecting subspaces in noiseless and noisy cases |
| $\ell_0$-induced Sparse Subspace Clustering ($\ell_0$-SSC) [128] | $\|\|C\|\|_0$ | all arrangements of noiseless distinct subspaces |
| Low Rank Representation (LRR) [59] | $\|\|C\|\|_*$ | noiseless independent subspaces |
| Least Square Regression (LSR) [66] | $\|\|C\|\|_F^2$ | noiseless independent subspaces |
| Multi-Subspace Representation (MSR) [69] | $\|\|C\|\|_1 + \lambda\|\|C\|\|_*$ | Not available |
| Subspace Segmentation via QP (SSQP) [115] | $\|\|C^\top C\|\|_1$ | noiseless orthogonal subspaces |
| Correlation Adaptive Subspace Segmentation (CASS) [64] | $\sum_j \|\|X \operatorname{diag}\big(C(:,j)\big)\|\|_*$ | noiseless independent subspaces |
| Elastic Net Subspace Clustering (ENSC) [134] | $\|\|C\|\|_1 + \lambda\|\|C\|\|_F^2$ | independent and disjoint subspaces for the noiseless case with strong dependency on the value of the parameter $\lambda$ |

to be subspace preserving, under some conditions; see Table 1. These conditions depend on the separation between the subspaces and the distribution of the data points within each subspace. With sufficiently separated subspaces and well-spread data points (not skewed towards a specific direction), sparsity regularization is guaranteed to provide subspace preserving coefficients in noiseless and noisy cases. For detailed theoretical discussions on this topic, we refer the interested reader to [21, 116].

Intersecting/overlapping subspaces is the most general subspace arrangement for which there is no particular assumption on the subspaces, and any two subspaces can have a nontrivial intersection [116, 95]. Note that data points belonging to the intersection of two subspaces lead to non-unique membership assignments. Similar to disjoint subspaces, sparsity is the only regularization that is proven to be effective for clustering intersecting subspaces. Two distinct two-dimensional planes in three dimensions is an example of intersecting subspaces.

The two most widely used algorithms for self-expressive based linear SC are the following:

- Sparse Subspace Clustering (SSC) [21] uses the component-wise $\ell_1$ norm, that is, $||C||_1 = \sum_{i,j} |C_{i,j}|$, as a convex surrogate of the $\ell_0$ norm to enhance the sparsity of $C$,
- Low-Rank Representation (LRR) [59] uses the nuclear norm, that is, $||C||_* = \sum_i \sigma_i(C)$ where $\sigma_i(C)$ is the $i$th singular value of $C$, to promote $C$ to have low rank.

SSC is the pioneer work in this context, and has strong theoretical guarantees in noisy and noiseless cases for independent, disjoint and intersecting subspaces [21, 95, 96, 136]. LRR has shown competing results with SSC. However, its behavior in noisy and disjoint subspaces is still not well understood theoretically[1] [117].

**Remark 2** (Oversegmentation). *Accurate SC not only depends on subspace preserving representations but also on the connectivity of data points within the same subspace. Even though sparsity regularization has strong theoretical guarantees compared to other regularizations, it not only emphasizes the sparsity of between-cluster connections, but also the sparsity of the inner-cluster connections. Hence, it is possible that the data points from the same subspace form multiple connected components [73], which leads to the creation of more clusters than necessary, that is, oversegmentation. The connectivity issue is less apparent in other regularizations as they inherently promote denser representations.*

### 2.1.2 Post-Processing

In order to enhance the quality of the coefficient matrix $C$, several post-processing strategies exist. The goal of these approaches is to decrease the number of wrong between-subspaces connections and/or strengthening the within-subspaces connections. Even though there is no generally accepted strategy in the literature, a few common post-processing methods are as follows:

1. Normalizing the columns of the coefficient matrix as $C(:,j) \leftarrow C(:,j)/||C(:,j)||_\infty$ where $||\cdot||_\infty$ is the infinity norm [21]. This can be helpful when some data points have drastically different norms compared to other data points.

2. Applying a hard thresholding operator on each column of the coefficient matrix by keeping only the $k$ largest entries in absolute value [84]. This post-processing is based on the property of Intra-subspace Projection Dominance (IPD) which states that the entries corresponding to data points in the same subspace are larger (in absolute value) than the ones corresponding to data points in different subspaces.

3. Performing an ad-hoc post-processing strategy using multiple steps. First, a percentage of the top entries in each column of the coefficient matrix are preserved. Next the shape interaction matrix method [15] is applied to remove the noise; it consists in a low-rank approximation of $C$ based on the skinny singular value decomposition. Finally each entry is raised to some power larger than one (the value 4 is the often used) in order to intensify the dominant connections. This strategy depends on several parameters with no explicit theoretical justification. However, it is widely used in nonlinear SC approaches based on neural networks [41, 42].

4. Finding *good* neighbors which correspond to the key connections in the coefficient matrix [125]. This approach is based on three parameters: $k > q > p$. First, only the $k$ largest coefficients are kept for each data point. Then, only a subset of these $k$ connections is preserved based on the notion of good neighbors. A good neighbor is defined as a data point with at least $p$ (usually $p = 1$) common data points inside the $k$ largest connections. The $q$ good neighbors of each data point with the largest coefficients form the final selected subset. A disadvantage of this approach is that it depends on parameters which are difficult to tune. (Note that using $q = k$ and $p = 0$ reduces to the second post-processing approach described above.)

[1]Although the nuclear norm is a convex surrogate of the rank, it has different subspace preserving properties; see [66, Example 1] for a numerical example. This is in contrast to the strong theoretical guarantees of the $\ell_0$ norm regularizations and its corresponding convex surrogate, the $\ell_1$ norm; see Table 1.

## 2.2 Spectral clustering using the coefficient matrix

After obtaining the coefficient matrix $C$ using (2), the next step is to infer the clusters using spectral clustering [94, 75]; see [110] for a tutorial. In fact, the entries in the coefficient matrix $C$ can be interpreted as the links between the data points: $C(i,j) \neq 0$ means that the data point $j$ is used for expressing data point $i$, and hence it is likely that the data points $i$ and $j$ belong to the same subspace. Therefore, the coefficient matrix $C$ corresponds to a directed graph structure $G = (V, E)$ where the nodes of the graph $(V)$ are the data points and the weights of the edges $(E)$ are determined by the entries in $C$. Using this interpretation, the problem of linear SC is mapped into the problem of segmenting the graph $G$. The symmetric adjacency matrix $A$ is constructed as $|C| + |C|^T$ and then the celebrated algorithm of spectral clustering is applied on the affinity matrix $A$ to partition the graph.

An important advantage of spectral based approaches is that they do not need to know the dimensions $(\{d_i\}_{i=1}^{c})$ of the subspaces. Moreover, by utilizing spectral clustering, they can estimate the number of clusters by analyzing the spectrum of the Laplacian matrix corresponding to the adjacency matrix $A$ [110]. Furthermore, robustness of spectral clustering to small perturbations is advantageous for *correct* clustering especially with unavoidable slight violation of subspace preservation in the coefficient matrix for real-world noisy cases [116].

Linear SC based on exploiting the self-expressive property initiated an extensive amount of research in various directions. Several extensions of self-expressive based SC approaches have been proposed in the past decade to overcome notable challenges such as scalability [72, 11, 2, 133], improving robustness [34, 65], multi-view data clustering [23, 52], and the ability to deal with missing data [54]. However, a crucial limit to linear SC is the linearity assumption. In the next section, we review the approaches that were proposed to segment data points that are drawn from nonlinear manifolds.

## 3 Nonlinear subspace clustering

Linear SC algorithms often fail in dealing with data belonging to several nonlinear manifolds. This is expected since these approaches only consider the *global* linear relationship between data points. For nonlinear manifolds, the *local* relationship between data points plays a more important role, as we will explain later in this section. But first, let us define the problem of nonlinear subspace clustering [20].

**Definition 6** (Nonlinear subspace clustering (nonlinear SC))**.** *Let $X \in \mathbb{R}^{d \times n}$ be the input data. Suppose the data points lie on $c$ manifolds $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_c$ with intrinsic dimensions*[2] *$d_1, d_2, \dots, d_c$ such that $d_i \ll d$ for all $i \in \{1, \dots, c\}$. The problem of nonlinear SC is defined as segmenting the data points based on their corresponding manifolds and obtaining a low-dimensional embedding of the data points within each manifold.*

Nonlinear SC is also referred to as *manifold clustering* in the literature [20]. We mainly use the nonlinear SC term to emphasize that this survey focuses on the nonlinear approaches that extend the concepts in linear SC for data on nonlinear manifolds. Nonlinear SC approaches can be divided into three main categories:

1. locality preserving: they exploit the local geometric structure of the manifold to bridge the gap between *thinking globally*, that is, using the global information from the whole data set, and *fitting locally*, that is, using spatial local information around each point.

[2] A manifold has dimension $d$ if every data point has a neighborhood homeomorphic to the Euclidean space in $\mathbb{R}^d$. For example, a circle is a one-dimensional manifold as the neighborhood of each point is locally a segment [18].

2. kernel based: they use implicit but predefined nonlinear mappings to transfer the data into a space where the linearity assumption is more likely to be satisfied.

3. neural network based: they use the recent advances in structured neural networks to learn a nonlinear mapping of the data that respects the union of linear SC structure.

Figure 2 illustrates these three categories, along with several subcategories. In the following three sections, the major approaches within each category are presented.

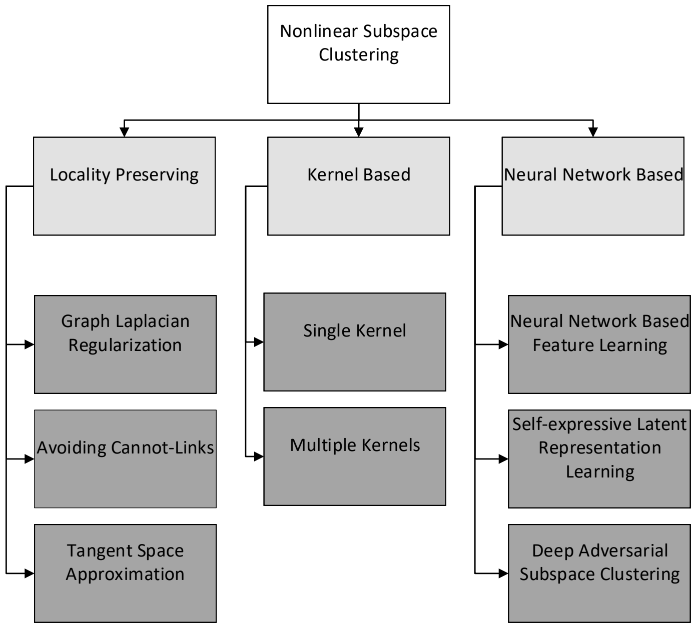

Nonlinear Subspace Clustering
Locality Preserving
Kernel Based
Neural Network Based
Graph Laplacian Regularization
Avoiding Cannot-Links
Tangent Space Approximation
Single Kernel
Multiple Kernels
Neural Network Based Feature Learning
Self-expressive Latent Representation Learning
Deep Adversarial Subspace Clustering

Figure 2: Categorization of nonlinear SC approaches.

## 3.1 Locality preserving nonlinear subspace clustering

A major disadvantage of linear SC approaches is that they are not *faithful* to the data structure in the high-dimensional space. In particular, the self-expressiveness property does not guarantee that the nearby points in the ambient space have similar representations in the latent coefficient space [114], that is, a small value of $\|X(:,i) - X(:,j)\|$ does not imply that $\|C(:,i) - W(:,j)\|$ is small. However, preserving the local configuration of the nearest neighbors of each point in the original ambient space is essential in being faithful to the data structure in nonlinear manifolds [90]. In order to reflect the local structures, there are three major categories: graph Laplacian regularization, avoiding cannot links, and tangent space approximation.

In fact, the root of these approaches comes from the single manifold learning algorithms based on preserving locality relationships among data points [91, 90, 6]. Inspired by these classical algorithms, two assumptions are needed for these approaches to work (although they are not always stated explicitly):

(i) the underlying manifolds are *smooth*, and

(ii) the manifolds are *well-sampled*, that is, there are sufficiently many data points sampled in the neighborhood of each data point.

Subsequently, these two assumptions imply that that each data point and the nearby points on the same manifold lie approximately on a *local linear patch*[3]. Hence, the nonlinear structure of the manifolds can be captured by the *local linear* reconstruction of the data points. Let us now present the approaches within each of the three categories in the following sections.

### 3.1.1 Graph Laplacian regularization

The methods in this category exploit the local structure of the data to construct the matrix $C$. To do so, a *pairwise* local similarity matrix $S \in \mathbb{R}^{n\times n}$ is first constructed from the data points. The matrix $S$ is symmetric, and its $(i,j)$-th entry measures how *similar* or *close* the data points $X(:,i)$ and $X(:,j)$ are in the ambient space. The entries of $S$ are computed in different ways, including the following:

1. Binary k-nearest neighbors (K-NN): $S(i,j)$ is equal to one if $i$ is among the $k$ nearest neighbor of $j$, or vice versa [37], that is,
$$S(i,j) = \begin{cases} 1 & \text{if } X(:,j) \in \mathcal{N}_k(X(:,i)) \text{ or } X(:,i) \in \mathcal{N}_k(X(:,j)), \\ 0 & \text{otherwise,} \end{cases}$$
where $\mathcal{N}_k(x)$ computes the $k$ nearest samples of the data point $x$.

2. Continuous similarity functions: the value $S(i,j)$ is computed using various similarity functions, such as Gaussain kernels [129, 67, 139] given by
$$S(i,j) = e^{\frac{-||X(:,i)-X(:,j)||_2^2}{\sigma^2}},$$
where $\sigma$ is a parameter. Other data dependent similarity functions such as the geodesic distance for data on Grassmann manifolds can also be used [112].

3. Weighted K-NN is a combiantion of the two strategies described above [60]:
$$S(i,j) = \begin{cases} \text{sim}(X(:,i), X(:,j)) & X(:,j) \in \mathcal{N}_k(X(:,i)) \text{ or } X(:,i) \in \mathcal{N}_k(X(:,j)), \\ 0 \quad \text{otherwise,} \end{cases}$$
where $\text{sim}\big(x,y\big)$ is a similarity function, such as Gaussian kernels, between two input vectors $x$ and $y$.

For a recent comprehensive overview of similarity and neighborhood construction algorithms, we refer the interested reader to [85].

Once the matrix $S$ is constructed, the following regularizer for $C$ is constructed using this local similarity information:
$$\mathcal{L}(C) = \frac{1}{2}\sum_{i,j} ||C(:,i) - C(:,j)||_2^2 \; S(i,j). \tag{3}$$

[3]In single manifold learning, it is proven that under the smoothness and well-sampled assumptions, for a *d-dimensional* manifold, each data point along with its $2d$ neighbors define an approximately linear patch of the manifold [91].

Let the Laplacian matrix $L \in \mathbb{R}^{n\times n}$ corresponding to $S$ be defined as $L = D - S$ where $D$ is the diagonal matrix with $D_{ii} = \sum_j S(i,j)$ for all $i$. The function (3) can be rewritten as:

$$\begin{aligned}\mathcal{L}(C) = \frac{1}{2}\sum_{i,j} ||C(:,i) - C(:,j)||_F^2 \; S(i,j) &= \sum_i C(:,i)^\top C(:,i) D_{ii} - \sum_{i,j} C(:,i)^\top C(:,j) S(i,j) \qquad (4)\\ &= \operatorname{tr}(CDC^\top) - \operatorname{tr}(CSC^\top) = \operatorname{tr}(CLC^\top),\end{aligned}$$

where $\operatorname{tr}(\cdot)$ denotes the trace of a matrix. This regularizer $\mathcal{L}(C)$ is known as the Graph Laplacian or manifold regularization. It is used within existing global linear SC optimization problems, that is, it is added in the objective function of the optimization problem (2) with a proper penalty parameter. Manifold regularization promotes a *grouping effect*, which is defined as follows.

**Definition 7** (Grouping Effect [37]). *Given the input data matrix $X \in \mathbb{R}^{d\times n}$, let $C \in \mathbb{R}^{n\times n}$ be the obtained coefficient matrix corresponding to $X$ by an SC approach. The matrix $C$ has the grouping effect if $||C(:,i) - C(:,j)||_2$ goes to zero as $X(:,i)$ and $X(:,j)$ get closer, that is, as $||X(:,i) - X(:,j)||_2$ goes to zero.*

Intuitively, the grouping effect encourages locally similar points (that is, pairs of points whose corresponding value in $S$ is large) to have similar coefficient representations.

The grouping effect of Graph Laplacian regularizers generates graphs that better represent the structure of the data and hence are better connected [37]. This might benefit SC approaches based on self-expressiveness as they use spectral clustering for the final step [110]. Note that the Graph Laplacian regularization is based on the assumption that the representation coefficient vector corresponding to each sample, that is, $C(:,i)$ for each $i = 1, ..., n$, is changing *smoothly* on each manifold [7]. Hence, nearby data points in the ambient space have similar coefficient vectors. This regularization also implicitly assumes that the data points from different clusters are not likely to be near each other and that sufficiently many data points are sampled from each manifold.

Graph Laplacian regularization can be used exclusively with the reconstruction error, $||X - XC||_F^2$, as in the Smooth Representation (SMR) algorithm [37]. SMR solves the following optimizatin problem:

$$\min_C \; \lambda ||X - XC||_F^2 + \operatorname{tr}(CLC^\top).$$

Note that SMR does not enforce $\operatorname{diag}(C) = 0$ explicitly, because the regularization $\operatorname{tr}(CLC^\top)$ prevents the identity matrix to be an optimal solution, for $\lambda$ sufficiently large, since the identity matrix $C = I$ satisfies $||C(:,i) - C(:,j)||_2 = 1$ for all $i \neq j$.

It is proven that Graph Laplacian regularization of the coefficient matrix leads to subspace preserving coefficients for noiseless linear independent subspaces [37]. Graph Laplacian regularization can be used in combination with other coefficient regularizations such as the $\ell_1$ norm [129], or the nuclear norm [60, 131]. In particular, using the $\ell_1$ norm along with Graph Laplacian regularization leads to the algorithm referred to as Laplacian Regularized $\ell_1$-SSC (LR$\ell_1$-SSC) which solves the following optimization problem [129]:

$$\min_C \; ||C||_1 + \frac{\lambda_1}{2} ||X - XC||_F^2 + \lambda_2 \operatorname{tr}(CLC^\top) \quad \text{such that } \operatorname{diag}(C) = 0.$$

However, as we will see in the numerical experiments in Section 5, promoting the smoothness of the representation over the manifold might not be sufficient to recover the complex nonlinear structures. Moreover, these methods are rather sensitive to the choice of the similarity function used to construct the pairwise local similarity matrix $S$.

### 3.1.2 Avoiding cannot-links

Instead of encouraging locally nearby points to have similar coefficient representations, as presented in the previous section, one could encourage the entries of $C$ corresponding to *faraway* points to have small/zero values. In other words, data points should not use faraway points in their representation, which we refer to as avoiding cannot-links. The name "cannot-links" is borrowed from the graph learning and spectral clustering literature to indicate the pairwise constraints on the links that are encouraged/enforced to be avoided [68, 119].

Avoiding cannot-links attracted less attention compared to the Graph Laplacian regularization methods. Generally, the cannot-links refer to graph links/coefficients corresponding to distant points that are specified based on a dissimilarity criterion. There are two strategies to integrate pairwise cannot-links constraints:

1. Explicit elimination of faraway samples: A simple approach to prevent cannot-links is to explicitly add them as proper constraints in the optimization problem; for example, in [155], Zhuang et al. use the following constraint: for all $j$,

   $$C(i,j) = 0 \qquad \text{for all } X(:,i) \notin \mathcal{N}_k\big(X(:,j)\big).$$

   A similar approach was proposed for linear SSC [31]. Instead of additional explicit constraints on the entries of the coefficient matrix, the self-expressiveness term is modified such that each sample is represented by the locally nearby data points.

2. Implicit penalizing of distant pairwise connections: Cannot-links information can be added as a weighted penalty term. For example, in [148], the following regularization term was proposed:

   $$\sum_{i,j} ||X(:,i) - X(:,j)||_2^2 \; |C(i,j)|.$$

   This term is similar to (3). However, for any two faraway data points $X(:,i)$ and $X(:,j)$, the corresponding $C(i,j)$ is encouraged to be small while this is not the case for graph Laplacian regularization (for which $S(i,j)$ will be close to zero). In other words, in graph Laplacian regularization, the nearby samples are encouraged to have similar coefficient representation but there is no explicit penalty on the coefficient representations corresponding to faraway data points. In fact, graph Laplacian regularization tend to produce dense coefficient matrices (see the numerical results in Section 5.1), hence do not have the disadvantage of oversegmentation that avoiding cannot-links may have [2].

   A similar idea was presented in [149], together with non-negativity constraint on the coefficient matrix $C$. Another penalty term to penalize "non-local" connectivities was proposed in [12], namely $\sum_{i,j} e^{\text{dist}(X(:,i),X(:,j))^2} C_{i,j}^2$, where dist is a normalized distance metric.

Avoiding non-local data points in self-expression might be intuitively similar to encouraging spatially close connections, as with the Graph Laplacian regularization. However, they lack the grouping effect of Graph Laplacian regularizations.

### 3.1.3 Tangent space approximation

The approaches in this category rely on the estimation of the local tangent space for each data point, by fitting an affine subspace in the neighborhood of each data point. A simple strategy to approximate such tangent spaces is to consider a fixed predefined number of nearest neighbors for each sample, and approximate the tangent space by fitting an affine subspace to the data point and its neighbors. The crucial question is how to define the neighborhood of each data point in order to balance two goals: large enough to capture the local geometry of the manifolds, and small enough to avoid data points from other manifolds/clusters so as to preserve the curvature information of the underlying manifold.

Despite the fact that choosing a *fixed* value for neighborhood size with no prior knowledge is challenging, Deng et al. [19] proposed a multi-manifold embedding approach based on the angles between approximated tangent spaces using a fixed neighborhood size. In particular, they assumed that if two data points belong to the same manifold, the angle between their corresponding tangent space is small whereas the angle is large if they belong to different manifolds. In contrast to estimating tangent space using fixed neighborhood size, a different strategy is to allow the size of the neighborhood to be arbitrary. This can be achieved via sparse representations in order to estimate the neighboring samples and simultaneously calculate the contribution weight of each neighborhood sample. These weights would be helpful to reflect the clustering information as well. In this section, we focus on this particularly effective strategy.

**Sparsity based tangent estimation for manifold clustering** In order to obtain the neighborhood of each point, Elhamifar and Vidal [20] assumed that the tangent space of the manifold it belongs to can be approximated by the low-dimensional affine subspace constructed using a few neighboring points from the same manifold. Based on this assumption, they proposed an approach dubbed Sparse Manifold Clustering and Embedding (SMCE). It relies on two main principles, *sparsity* and *locality*, to estimate the tangent spaces.

For an illustration of the main idea behind SMCE, consider the example in Figure 3, showing two clusters of samples (gray circles) and their underlying manifolds. Suppose we want to approximate the tangent space for the sample $x_1$. Based on the principle of sparsity, the sparsest affine representation for the sample $x_1$ should contain two other distinct samples and among all possible combinations of two samples, $x_1$ is close to the affine span of $\{x_2, x_7\}$ and $\{x_2, x_3\}$. To avoid representing $x_1$ using the affine subspace containing $x_7$, which belongs to the other manifold, the locality principle comes into effect to favor the affine span of $x_2$ and $x_3$. However, it should be noted that locality alone is not helpful to avoid representing $x_1$ using the points $\{x_4, x_5, x_6\}$ which are spatially close to $x_1$. This example illustrates the necessity of both sparsity and locality in the approximation of local tangent spaces using the samples from the same manifold.

More precisely, SMCE works as follows. To construct the tangent space to $X(:,i)$, the normalized sample differences is first constructed as follows:

$$U_i = \left[\frac{X(:,1)-X(:,i)}{||X(:,1)-X(:,i)||_2}, \dots, \frac{X(:,i-1)-X(:,i)}{||X(:,i-1)-X(:,i)||_2}, \frac{X(:,i+1)-X(:,i)}{||X(:,i+1)-X(:,i)||_2}, \dots, \frac{X(:,n)-X(:,i)}{||X(:,n)-X(:,i)||_2}\right].$$

Based on these directions, the affine subspace $\mathcal{F}_i$ is represented using

$$\mathcal{F}_i = \left\{X(:,i) + U_i c_i \,|\, e^\top c_i = 1, c_i \in \mathbb{R}^{n-1}\right\},$$

where $e$ is the all-one vector. The goal is to estimate parameters of the local subspace $\mathcal{F}_i$ such that it has minimal distance to the unknown target tangent space. Hence, the coefficients $c_i$ in the linear combination are calculated such that the distance between $X(:,i)$ and the affine subspace $\mathcal{F}_i$ is minimized:

$$\min_{c_i} ||X(:,i) - \big(X(:,i) + U_i c_i\big)|| \; = \; \min_{c_i} \; ||U_i c_i||.$$

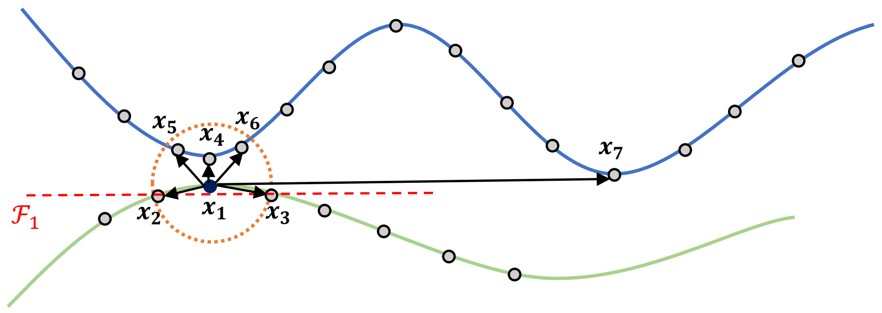


Figure 3: Illustration of the necessity of sparsity and locality in manifold clustering. Local affine directions through sample $x_1$, that is, $x_2 - x_1, \ldots, x_7 - x_1$ are shown with arrows. The dotted circle is the smallest ball centered at $x_1$ and containing $x_2$ and $x_3$ from the same manifold. Sparsity helps to avoid approximating the tangent space by close samples such as $x_4, x_5$ and $x_6$. Locality excludes the use of $x_7$ which belongs to the other cluster. The tangent space computed by SMCE is $\mathcal{F}_1$.

However, this formulation does not guarantee that the data points are expressed using locally nearby points from the same manifold. To estimate the tangent space using few nearby samples, SMCE optimizes the following problem for each point $X(:,i)$:

$$\min_{c_i} ||Q_i c_i||_1 + \frac{\lambda}{2}||U_i c_i||_2^2 \quad \text{such that } e^\top c_i = 1, \tag{5}$$

where $Q_i \in \mathbb{R}^{(n-1)\times(n-1)}$ is called the proximity inducing matrix. The matrix $Q_i$ is a positive-definite diagonal matrix. The entries of $Q_i$ are defined by normalized $\ell_2$ distance between each data point $X(:,j)$ and $X(:,i)$ for $j \neq i$, that is,

$$Q_i(j,j) = \frac{||X(:,j) - X(:,i)||_2}{\sum_{t\neq i} ||X(:,t) - X(:,i)||_2}.$$

Hence the diagonal entries of $Q_i$ that correspond to closer samples to $X(:,i)$ have smaller values, allowing the corresponding entries of $c_i$ to be larger. In contrast, the diagonal entries of $Q_i$ which correspond to farther samples have larger values, hence favoring smaller values of $c_i$. The nonzero entries of the vector $c_i$ indicate the points that are estimated to be on the same manifold as $X(:,i)$. Finally, the coefficient matrix $C$ is constructed as follows: $C(i,i) = 0$ for all $i$, and

$$C(i,j) = \frac{\frac{c_i(j)}{||X(:,j)-X(:,i)||_2}}{\sum_{t\neq i} \frac{c_i(t)}{||X(:,t)-X(:,i)||_2}} \quad \text{for} \quad j \neq i. \tag{6}$$

Applying spectral clustering on the corresponding symmetrized affinity matrix of $C$ reveals the final manifold clustering assignments.

A very similar approach to SMCE was presented in [144]. However, the authors enforced locality and sparsity in separate steps. First a sparse representation of the data points using the affine directions is computed (without the proximity inducing matrix), and then the obtained sparse representation is weighted using the spatial locality information.

### 3.2 Kernel based non-linear subspace clustering

Kernel methods are commonly used to identify nonlinear structures and relationships among data points. They map the data from the input space $\mathcal{X}$ to the reproducing kernel Hilbert space $\mathcal{H}$ where the points might be better represented for a specific task. In nonlinear SC, the goal is to map the data points through a nonlinear transformation such that the linearity assumption is better satisfied; see Figure 4 for a simple illustration. However, instead of explicitly using a nonlinear function, Kernel

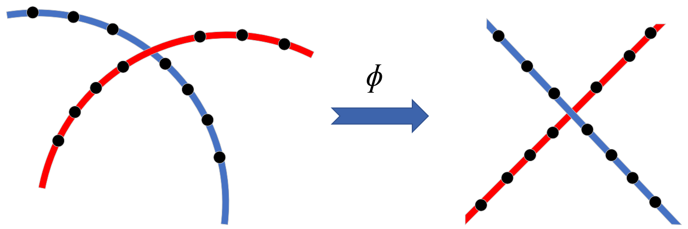


Figure 4: Using implicit nonlinear transformations $\phi$ in kernel based approaches, the data is mapped to a space which might be more suitable for linear subspace clustering.

methods rely on the kernel trick to implicitly apply the nonlinear transformation. The kernel trick represents the data through pairwise similarity functions. It avoids the often computationally expensive nonlinear transformations (feature maps), that is, the explicit computation of the coordinates of the transformed data points in the feature space. Using this trick, only the inner products between all pairs of transformed data points in the feature space are computed.

Let $\phi : \mathcal{X} \to \mathcal{H}$ be the explicit nonlinear feature map. The kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for $x, y \in \mathbb{R}^d$ is defined as: $k(x, y) = \langle \phi(x), \phi(y) \rangle$, where $\langle ., . \rangle$ is the inner product. The matrix $\mathcal{K}_{XX} \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix, known as the Gram matrix, whose entries are defined as:

$$\mathcal{K}_{XX}(i, j) = k(X(:, i), X(:, j)) = \langle \phi(X(:, i)), \phi(X(:, j)) \rangle. \tag{7}$$

Using the kernel function, the explicit representation for the nonlinear transformation $\phi$ is no longer needed.

In this section, we provide an overview of existing nonlinear SC based on kernel methods. These approaches can be divided into two categories: single kernel and multiple kernels, and are presented in the next two sections.

#### 3.2.1 Using a single kernel

Patel et al. [79] proposed a kernelized SC approach by using the kernel trick in the (affine) sparse SC formulation. Their algorithm, dubbed Kernelized Sparse Subspace Clustering (KSSC), solves the following optimization problem

$$\min_{C} ||C||_1 + \frac{\lambda}{2} ||\Phi(X) - \Phi(X)C||_F^2 \quad \text{such that} \quad \text{diag}(C) = 0, \; C^\top e = e, \tag{8}$$

where $\Phi(X) = [\phi(X(:,1)), \phi(X(:,2)), \dots, \phi(X(:,n))]$. The kernel trick can be used to avoid computing $\Phi(X)$ explicitly, using

$$\begin{aligned} ||\Phi(X) - \Phi(X)C||_F^2 &= \operatorname{tr}\left(\Phi(X)^\top \Phi(X)\right) - 2\operatorname{tr}\left(\Phi(X)^\top \Phi(X) C\right) + \operatorname{tr}\left(C^\top \Phi(X)^\top \Phi(X) C\right) \\ &= \operatorname{tr}\left(\mathcal{K}_{XX} - 2\mathcal{K}_{XX}C + C^\top \mathcal{K}_{XX} C\right), \end{aligned} \tag{9}$$

where the kernel $\mathcal{K}_{XX}$ is given in (7). As for SSC, the authors used sparse regularization for the coefficient matrix. Finally, (8) can be reformulated as follows

$$\min_C ||C||_1 + \frac{\lambda}{2} \operatorname{tr}\left(\mathcal{K}_{XX} - 2\mathcal{K}_{XX}C + C^\top \mathcal{K}_{XX} C\right) \quad \text{such that} \quad \operatorname{diag}(C) = 0,\ C^\top e = e. \tag{10}$$

In order to reduce the computational cost of calculating the coefficient matrix for high-dimensional data, the same authors [78] extended KSSC to combine dimensionality reduction and kernelized SC. To this end, their approach learns the implicit mapping of the data onto a high-dimensional feature space using the kernel method, and then projecting onto a low-dimensional space via a linear projection.

The ease of use of the kernel trick in the self-expressive based linear SC initiated the development of other nonlinear SC algorithms with additional regularizations, motivated by different purposes. These algorithms include for example the kernelized multi-view SC approaches from [138, 121]. Recently Kang et al. [48] proposed additional structure preserving regularization for KSSC in order to minimize the inconsistency between inner products of the transferred data, $\Phi(X)$, and inner products for reconstructed transferred data, $\Phi(X)C$.

However, the traditional challenges of kernel methods are still present for kernel based SC. The two main challenges are:

1. Using the Frobenius norm to compute the representation error facilitates the use of the kernel trick; see (9). However, the Frobenius norm implicitly assumes that the noise follows a Gaussian distribution. How can the kernel trick be applied on non-Gaussian noise models including, e.g., gross corruptions and outliers?

2. With no prior knowledge, how can we select an appropriate kernel function and its parameters?

In the rest of this section, we review the few approaches that tried to tackle these challenging limitations.

**Robust kernelized nonlinear SC** To improve the robustness, Xiao et al. [120] adopted the $\ell_{2,1}$ norm to replace the Frobenius norm in the kernelized LRR algorithm. The $\ell_{2,1}$ norm of the matrix $E \in \mathbb{R}^{d \times n}$ is defined as

$$||E||_{2,1} = \sum_{i=1}^{n} ||E(:,i)||_2 = \sum_{i=1}^{n} \sqrt{E(:,i)^\top E(:,i)}.$$

This norm, which enhances column-wise sparsity, models sample-specific outliers assuming that a fraction of the data points are corrupted. Based on this norm, *robust* nonlinear LRR solves the following optimization problem:

$$\min_C \ ||C||_* + \lambda ||\Phi(X) - \Phi(X)C||_{2,1}. \tag{11}$$

Defining $P = I - C \in \mathbb{R}^{n \times n}$, (11) can be reformulated as

$$\min_{P,C} ||C||_* + \lambda \sum_{i=1}^{n} \sqrt{P(:,i)^\top \Phi(X)^\top \Phi(X) P(:,i)}, \tag{12}$$

which allows to use the kernel trick. To the best of our knowledge, no kernelized SC algorithm is proposed for other robust noise measurement norms such as the component-wise $\ell_1$ norm.

**Adaptive kernel learning** All the aforementioned methods (and in fact, the majority of kernel based algorithms) make use of standard predefined kernel functions, such as Gaussian RBF kernels $k(x,y) = \exp\big(-\frac{||x-y||_2^2}{2\sigma^2}\big)$, and polynomial kernels $k(x,y) = \big(x^\top y + a\big)^b$. However, due to lack of a criterion for measuring the *quality* of a kernel function, selecting a proper kernel function is challenging. In the context of SC, a *good* kernel should result in a union of linear subspaces in the implicit embedded space. Following this intuition, Ji et al. [40] presented a kernel SC approach where they explicitly enforced the feature map $\Phi(X)$ to be of low rank and self-expressive:

$$\min_{C} ||\Phi(X)||_* + \lambda ||C||_1 \quad \text{such that} \quad \Phi(X) = \Phi(X)C, \quad \text{diag}(C) = 0, \quad C^\top e = e.$$

However, this formulation has two limitations: (i) applying the kernel trick on the first term of the objective function is not straightforward, and (ii) the lack of constraints on the kernel leads to trivial solutions such as mapping all data points to zero. In order to overcome these issues, the authors exploited the symmetric positive definiteness of the kernel gram matrix $\mathcal{K}$, which implies that $\mathcal{K} = B^\top B$ for some square matrix $B$, while

$$||\Phi(X)||_* = ||B||_* \quad \text{for all } B \text{ such that } \mathcal{K} = B^\top B. \tag{13}$$

Moreover, they restricted the unknown kernel matrix to be close (but not identical) to a predefined kernel matrix (such as Gaussian RBF or polynomial kernels) to avoid trivial solutions:

$$\begin{aligned} \min_{C,B,\Phi} \; & ||B||_* + \lambda_1 ||C||_1 + \frac{\lambda_2}{2} ||\Phi(X) - \Phi(X)C||_F^2 + \lambda_3 ||K_G - B^\top B||_F^2, \\ \text{such that} \quad & C^\top e = e, \quad \text{diag}(C) = 0, \quad \Phi(X)^\top \Phi(X) = B^\top B, \end{aligned} \tag{14}$$

where $K_G \in \mathbb{R}^{n \times n}$ is the predefined kernel matrix. Using the kernel trick (9) and substituting $\Phi(X)^\top \Phi(X)$ by $B^\top B$, they solved this problem using ADMM. This formulation ensures that the transformed data points in the latent feature space are low-rank while an *adaptive* kernel function is used. The optimization problem (14) was further extended in [122] by regularizing the mapped features $\Phi(X)$ using the *weighted Schatten p-norm* as a tighter nonconvex surrogate for the rank and using correntropy for more robust measurement of the difference between the predefined kernel $K_G$ and the estimated kernel $BB^\top$. The same idea was used in the robust nonlinear multi-view SC approach in [145].

However, these approaches suffer from two limitations: (i) they still need a predefined kernel matrix, $K_G$, and (ii) assuming that a "good" kernel leads to a low-rank embedding of the data points is rather strong for data on multiple manifolds. In particular, there is no guarantee that learning a low-rank kernel corresponds to an implicit nonlinear transformation that preserves sufficient separability between the data points from different manifolds. In other words, a low-rank kernel (even with the additional self-expressiveness structure) does not necessary lead to a well separated embedding of the data points.

### 3.2.2 Using multiple kernels

Even though kernel methods are considered principled approaches to provide non-linearity in linear models, they rely on selecting and tuning a kernel. In many (unsupervised) applications, with no prior knowledge, this is a challenging problem. Multiple Kernel Learning (MKL) methods overshadow this issue, by learning a consensus kernel $\mathcal{K}$ from a set of predefined candidate kernels $\{\mathcal{K}_G^{(i)}\}_{i=1}^k$, where $k$ is the number of given kernels. The sought kernel $\mathcal{K}$ is regularized towards the predefined candidate kernels via a penalty function $h\left(\mathcal{K}, \{\mathcal{K}_G^{(i)}\}_{i=1}^k, w\right)$ which is added as a penalty term in the objective function. The vector $w \in \mathbb{R}_+^k$ contains the weights/importance of the predefined kernels. This regularization function is typically defined in two ways:

1. Centroid-based MKL [49] encourages the consensus kernel $\mathcal{K}$ to be close to every predefined kernel by minimizing

$$h\left(\mathcal{K}, \{\mathcal{K}_G^{(i)}\}_{i=1}^k, w\right) = \sum_{i=1}^k w(i) \left\|\mathcal{K} - K_G^{(i)}\right\|_F^2 . \tag{15}$$

2. Linear combination MKL [89] encourages the consensus kernel to be approximately a convex combination of the predefined kernels:

$$h\left(\mathcal{K}, \{\mathcal{K}_G^{(i)}\}_{i=1}^k, w\right) = \left\|\mathcal{K} - \sum_{i=1}^k w(i)\mathcal{K}_G^{(i)}\right\|_F^2 \quad \text{where} \;\; w \geq 0, \; \sum_{i=1}^k w(i) = 1. \tag{16}$$

The main difference between MKL based approaches is the enforced property for the ideal consensus kernel, using different regularizations. This relates to the criteria used to define the "goodness" of the consensus kernel, and the two most widely used ones promote

- a block-diagonal coefficient matrix, or
- a low-rank consensus kernel.

Let us discuss these two strategies in the next two paragraphs.

**MKL encouraging a block-diagonal representation** A criterion for quantifying the quality of the unknown consensus kernel is based on the assumption that it should encourage a block diagonal representation for the coefficient matrix (up to permutation), where each block corresponds to a different manifold. In fact, the coefficient matrix $C$ is block diagonal, up to permutation, if and only if the subspace preserving property holds (see Definition 2).

A common approach to promote a block-diagonal representation is to use a proper regularizer for the coefficient matrix. Let us define the Laplacian matrix $L$ associated to $C$ as $L = \text{diag}(Ae) - A$ where $A = |C| + |C|^\top$ is the affinity matrix corresponding to $C$. Then, minimizing the sum of the $c$ smallest eigenvalues of $L$, that is, $\sum_{i=1}^c \sigma_i(L)$ where $\sigma_i(L)$ is the $i$-th smallest eigenvalue of $L$, promotes $C$ to be block diagonal. In fact, the multiplicity of the zero eigenvalue of the Laplacian matrix is equal to the number of connected components in the graph corresponding to the matrix $C$ [110]. Using the Ky Fan theorem, Lu et al [62] write this nonconvex regularizer into a convex counterpart.

As one of the first approaches presented for MKL based nonlinear SC, Kang et al. [49] proposed the Self-weighted Multiple Kernel Learning (SMKL) algorithm. SMKL combines the multi-kernel learning formulation in (15) with the block-diagonal regularizer in the kernelized LSR formulation [66][4]:

$$\min_{C,\mathcal{K}} \underbrace{||\Phi(X)-\Phi(X)C||_F^2+\lambda_1||C||_F^2}_{\text{kernel based LSR self-expressiveness}}+\underbrace{\lambda_2\sum_{i=1}^{k} w(i)||\mathcal{K}-K_G^{(i)}||_F^2}_{\text{Multiple kernel learning}}+\underbrace{\lambda_3\sum_{i=1}^{c}\sigma_i(L)}_{\text{block-diagonal}}, \tag{17}$$

$$\text{such that } \; C\geq 0,\; C=C^\top,\; \mathcal{K}=\Phi(X)^\top\Phi(X) \text{ and } L=\operatorname{diag}(Ce)-C,$$

where the kernel trick can be easily applied on the first term in the objective function; see (9). An iterative alternative optimization approach was used to solve (17). In each iteration, the entries of the weight vector $w$ are updated adaptively during the optimization process using

$$w(i)=\frac{1}{2||\mathcal{K}-K_G^{(i)}||_F^2}.$$

However, in our implementations, we noticed that it might prevent convergence, because the parameter $w$ in (17) is modified at each iteration. Moreover, with no constraint on the summation of the weights, they tend to have very small values.

**Remark 3** (Nonnegativity). *The nonnegativity and symmetry constraints in* (17) *make the coefficient matrix $C$ a valid affinity matrix [62], with $L=\operatorname{diag}(Ce)-C$ being the corresponding Laplacian matrix. However, with the nonnegativity constraint on the entries of the matrix $C$, the optimal solution may not be subspace preserving for the "extreme points" within the subspaces (since they are not contained within the conical hull of points from the same subspace). As discussed for affine SC in [56], the spectral clustering step is however likely to generate correct clusters. This is justified by the "collaborative effect" phenomenon which states that the coefficient representation corresponding to non-extreme points can "pull" the extreme points toward the data points corresponding to the same subspace, and hence make the spectral clustering robust to the wrong connections. However, this depends on the strength of the within subspace connectivity of non-extreme data points and the number of the extreme data points.*

Similar formulations were presented in [124, 147, 88]. In these approaches, the update of $w$ was performed using variations of the Correntropy Induced Metric (CIM) for measuring the distance between entries of the kernels, that is, $w(i)=CIM(\mathcal{K},K_G^{(i)})$. The CIM between two matrices $X\in\mathbb{R}^{N\times N}$ and $Y\in\mathbb{R}^{N\times N}$ is defined as $CIM(X,Y)=1-\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}G(X(i,j)-Y(i,j))$ where $G(x)=\exp(-x^2/2\sigma^2)$, and is known to be a more robust distance metric compared to the Frobenius norm [65], and hence might lead to a more accurate weighting assignment. In particular, [124] used the mere block-diagonal regularizer and eliminated the Frobenius norm regularizer for the coefficient matrix. This is motivated by the fact that block-diagonal regularizer is proven to be subspace preserving for independent subspaces (similar to the Frobenius norm) [63]. The same idea was used for multi-view nonlinear SC based on MKL in [147]. Ren et al. [88] introduced an additional regularization to learn the affinity and coefficient matrices simultaneously.

**MKL encouraging a low-rank consensus kernel** Although the data might be nonlinear in the original ambient space, the linear subspace structure should be present in the implicit embedding

[4]LSR is a linear SC algorithm that regularizes the coefficient matrix $C$ with the Frobenius norm; see Table 1.

space. Hence a proper kernel should implicitly lead to a low-rank embedded data. Based on this intuition, Kang et al. [51], proposed Low-rank Kernel learning for Graph matrix (LKG) that learns a *low-rank consensus kernel* from a weighted linear combination of the given kernels by solving the following optimization problem:

$$\min_{C,\mathcal{K},w} \underbrace{\frac{1}{2}||\Phi(X) - \Phi(X)C||_F^2 + \lambda_1 f(C)}_{\text{kernel based self-expressiveness}} + \underbrace{\lambda_2||\mathcal{K} - \sum\nolimits_{i=1}^{k} w(i)\mathcal{K}_G^{(i)}||_F^2}_{\text{Multiple kernel learning}} + \underbrace{\lambda_3\; ||\mathcal{K}||_*}_{\text{low-rank kernel}}\;, \tag{18}$$

$$\text{such that } \; C \geq 0,\; \mathcal{K} = \Phi(X)^\top \Phi(X) \geq 0,\; w \geq 0,\; \sum_{i=1}^{k} w(i) = 1.$$

However, [89] explained that minimizing $||\mathcal{K}||_*$ does not necessarily lead to a low-rank transformed data in the implicit feature space, that is, a small value for $||\Phi(X)||_*$. In fact, we have

$$||\mathcal{K}||_* = \operatorname{tr}\sqrt{\mathcal{K}^\top \mathcal{K}} = \operatorname{tr}\left(\Phi(X)^\top \Phi(X)\right) = ||\Phi(X)||_F^2.$$

Hence, using the observation in (13), and defining an auxiliary square matrix $B$ such that $\mathcal{K} = B^\top B$, they enforced the low-rankness of $\Phi(X)$ by minimizing the nuclear norm of $B$ [89].

## 3.3 Neural network based nonlinear SC

Neural networks have emerged as a very powerful representation learning framework and have drawn substantial attention due to many successful applications in different fields. The past four years have witnessed an increasing number of papers which tried to merge the representational power of neural networks with classical linear SC algorithms to deal with data from nonlinear manifolds. In a nutshell, the main motivation of neural networks based SC approaches is to overcome the fundamental limitation of kernel based alternatives, by *learning the proper embedding of the data from the data.* Geometrically, as for Kernel-based approaches, this proper embedding should ideally have a union of linear subspaces structure where classic linear self-expressiveness leads to subspace preserving representations. These algorithms can be partitioned into three main categories (see also Table 2, page 9):

1. **Neural network based feature learning** that extract nonlinear features from the data using neural networks as a preprocessing step for linear SC algorithms,

2. **Self-expressive latent representation learning** that enforce explicitly the neural network to learn self-expressive latent representation either by changing the architecture or the objective function, and

3. **Deep adversarial SC** that uses generative adversarial networks to perform nonlinear SC.

In the next three sections, we describe these three categories.

### 3.3.1 Neural network based feature learning

Early attempts for employing neural network for nonlinear SC were mainly limited to providing better, that is, *more discriminative*, features (representations) for the data points. In fact, several SC papers in the literature (including linear and nonlinear methodologies) still rely on manually-designed/handcrafted features, such as Local Binary Patterns (LBP) [77], Histogram of Oriented

Gradients (HOG) [17], and Scale-Invariant Feature Transform (SIFT) [61], for satisfactory results on real-world data sets. However, there is no theoretical justification on how any of these handcrafted features might affect the geometric structure of the data lying on multiple (nonlinear) subspaces. From the perspective of unsupervised feature learning, features extracted by neural networks have an important advantage over traditional handcrafted features: they are specifically *learned from the data*.

The focus of neural network based approaches for SC is to *learn* proper representations/features from the data points for the specific task of SC. The main shared characteristic of these approaches is that they treat feature learning and clustering as *separate tasks*, which is suboptimal. They are divided into two main subcategories: (i) transforming the unsupervised problem into a (self)-supervised problem, and (ii) using the connectivity information from the output of a classical linear SC algorithm as prior knowledge. Let us discuss, in the next paragraphs, the approaches within each category in detail.

**Self-supervised subspace clustering** The success of *supervised* feature learning in neural networks inspired a few works to map the unsupervised SC task to a (self-)supervised problem. To do so, the clustering output of a *linear* SC algorithm can be used as labels to provide supervision [92, 150]. However, these labels are highly unreliable as they are obtained by applying algorithms based on the linearity assumption. There are two strategies to exploit the information from these *noisy* labels to train a network:

- Use the concept of "confident learning": a confidence weight is assigned to each data point quantifying its likeliness of having the correct label [76]. The highly confident samples and their corresponding labels can be used to train a neural network in a supervised fashion. Sekmen et al. [92] used the criterion of the distance of the samples to the obtained subspaces from a linear SC algorithm to choose highly confident data points for the supervision. However, there are serious drawbacks to this approach. The weighting procedure based on the criterion of distance to the estimated subspaces using linear algorithms is not reliable for nonlinear data. Moreover, selecting the highly confident labels depends on the value of a threshold which is not easy to set with no prior knowledge. However, we would like to emphasize that confident learning is an emerging topic in robust deep learning given noisy labels with significant theoretical and numerical advances over the past couple of years [43, 70]. However, how to select the "confident samples" and use them for nonlinear manifold clustering task requires more in-depth study.

- Increase the number of clusters incrementally: To reduce the effect of wrong labels, Zhou et al. [150] suggested an iterative approach. Specifically, at the first iteration, the linear SSC algorithm is applied on the (raw) input data and the spectral clustering is applied on the coefficient matrix to segment the data in *two* clusters. A neural network is trained with the obtained labels (clusters) for a binary classification problem. In the following iterations, the features from the previously trained network are used as input data to the SSC and the number of clusters to segment the corresponding graph (in the spectral clustering step) is increased by one. The network parameters are updated using the obtained labels as a supervised classification problem. This iterative process continues until the number of clusters reaches the desired predefined number. However, there is no theoretical evidence or justification that this strategy can improve the robustness to the erroneous output of linear SC algorithms.

**Linear based connectivity as prior knowledge** Another strategy to leverage linear SC algorithms as a prior information is to use the coefficient matrix to guide the geometric structure of the embedded data points in the latent space of neural networks.

As the pioneer algorithm, Peng et al. [81, 83] proposed a structured autoencoder, dubbed StructAE, where self-expressiveness of the latent representations is incorporated in the loss function of the network. StructAE relies on a previously computed coefficient matrix $C$ by a classical linear SC algorithm. The authors suggested the use of SSC and LSR, leading to StructAE-L1 and StructAE-L2 respectively[5]. Chen et al. [13] used LRR instead to construct the matrix $C$. Given $C$, consider an autoencoder with $M$ fully-connected hidden layers, where the first $M/2$ layers indicate the encoder and the last $M/2$ layers indicate the decoder. We denote $\Theta$ the parameters of the network, that is, the weights and biases of each layer. Let us also denote the compact latent representation of the encoder output as $Z_\Theta(:,i)$ for the input data $X(:,i)$, and, the reconstructed data, which is the output of the decoder, by $\tilde{X}_\Theta$. They both depend on the parameters $\Theta$.

The optimization problem is defined as follows

$$\min_{\Theta} \frac{1}{2}||X - \tilde{X}_\Theta||_F^2 + \frac{\lambda_1}{2}||Z_\Theta - Z_\Theta C||_F^2 + \frac{\lambda_2}{2}||\Theta||_F^2, \tag{19}$$

where the first term is the reconstruction error of the autoencoder, the second term enforces the latent representation $Z$ to respect the multiple subspace structure encoded by the matrix $C$, and the third term is regularizing the parameters of the network to avoid overfitting. The coefficient matrix $C$ contains prior information on the *global structure* of the data, because it is obtained by minimizing $||X - XC||_F^2$, via a linear SC approach, containing the information of all data points using self-expressiveness. The clustering membership is obtained by clustering the latent representation $Z$ using algorithms such as k-means or by applying a linear SC algorithm such as SSC and LSR. The main motivation of StructAE is to integrate the individual and global structures together, and encourage the autoencoder to learn features/latent representation respecting the geometric structure provided by the coefficient matrix. However, the matrix $C$ is obtained by a *linear* SC based on a global linearity assumption. Hence the performance tends to significantly depend on how well the linearity assumption is satisfied.

### 3.3.2 Self-expressive latent representation learning

Separating the two modules of representation/feature learning and (subspace) clustering assignment typically leads to suboptimal results. To overcome this limitation, a group of approaches explicitly include the self-expressiveness term in the network loss function. These approaches are based on the intuition that minimizing self-expressiveness in the embedded space encourages the neural network to learn the *appropriate* embedding for the task of SC. In general, the optimization problem of these networks has the following form:

$$\min_{C,\Theta} \frac{1}{2}||Z_\Theta - Z_\Theta C||_F^2 + \lambda_1 f(C) + \lambda_2 h(X, Z_\Theta, C) \quad \text{such that} \quad Z_\Theta = \Phi_e(X, \Theta) \text{ and } \operatorname{diag}(C) = 0, \tag{20}$$

where $\Phi_e$ and $\Theta$ denote the network embedding and parameters, respectively. The embedding of the network, which is denoted by $Z_\Theta$, is encouraged to have a union of linear subspaces structure by explicitly minmizing the self-expressive representation term. As in the rest of the paper, $f(C)$ is the regularization on the coefficient matrix. The extra regularization $h(X, Z_\Theta, C)$ plays a critical role in

[5]Recall that SSC and LSR use the $\ell_1$ and $\ell_2$ norms for regularization of the columns of the coefficient matrix, respectively; see Section 2.1.1.

removing trivial solutions and specifying the properties of the nonlinear mapping and the embedding space. In particular, looking for self-expressiveness in the transformed space is not sufficient. There exists infinite nonlinear transformations that can lead to small/zero self-expressive error, that is, $||Z_\Theta - Z_\Theta C||_F \approx 0$. Hence, the main difference between these approaches lies on the regularization function $h$ that reduces the possible nonlinear mappings space and forms the geometric structure of the data embedding. There are mainly four choices for $h$ in the literature:

1. Instant normalization [82] promotes the norm of the embedded data points in the latent space to be close to 1 using the regularization $\frac{1}{2}\sum_{i=1}^{n} ||Z_\Theta(:,i)^\top Z_\Theta(:,i) - 1||_2^2$. This avoids an arbitrary scaling factor in the embedding space.

2. Locality preservation [153] captures the local structure of data through Graph Laplacian regularization, similarly as done in Section 3.1.1. This regularization is defined as $\frac{1}{2}\operatorname{tr}(CLC^\top)$, where $L$ is the Laplacian matrix defined from a similarity matrix over the data points $X$.

3. Autoencoder reconstruction loss [42] is based on an autoencoder architecture and uses a decoder network to reconstruct the original data from the self-expressive representation, that is, $Z_\Theta C$. More precisely, the regularization is defined as $||X - \tilde{X}_\Theta||_F^2$ where $\tilde{X}_\Theta = \Phi_d(ZC, \Theta)$, and $\Phi_d$ represents the decoding mapping which depends on the parameters $\Theta$. This regularization that minimizes the reconstruction error of the input matrix ensures that the self-expressive based embedding preserves sufficient information to recover the original data.

4. Restricted transformations [71] is based on learning stacked linear transformations through multiple layers that are connected via non-negativity constraints inspired by the rectified linear unit (ReLu) that sets negative values to zero after each layer [30, 25]. Specifically, Maggu et al. [71] proposed a three layered transformation for SC, solving the following optimization problem:

$$\min_{\Theta=\{T_i\}_{i=1}^3, C} \quad \underbrace{||T_3T_2T_1X - Z_\Theta||_F^2}_{\text{deep transformed data: } Z_\Theta \approx \Phi_e(X,\{T_i\}_{i=1}^3)} + \underbrace{\lambda_1||Z_\Theta - Z_\Theta C||_F^2 + f(C)}_{\text{self-expressiveness in latent space}} \tag{21}$$
$$\underbrace{+\lambda_2 \sum_{i=1}^{3}\big(||T_i||_F^2 - \operatorname{logdet} T_i\big)}_{\text{avoid trivial transformations}}, \quad \text{such that} \quad \underbrace{T_2T_1X \geq 0 \text{ and } T_1X \geq 0}_{\text{nonnegative representations}},$$

where $\{T_i\}_{i=1}^3$ are the linear transformations such that the dimension of each layer output is reduced. To avoid trivial or degenerate solutions ($T_i \to 0$ or $T_i \to \infty$), the linear transformation corresponding to each layer is penalized by minimizing the Frobenius norm minus the logarithm of the determinant of the transformation matrices. Note that the non-negativity constraints are inspired by the ReLU activation but they are inherently different.

Undoubtedly, the deep SC approaches based on autoencoder regularization are the most popular. A large number of algorithms have been proposed using this regularization. In the rest of this section, we focus on these approaches.

**Autoencoder regularized deep subspace clustering** The autoencoder based feature extraction for SC was first proposed as the Cascade Subspace Clustering (CSC) algorithm in [80]. In a nutshell, CSC first learns compact features by pretraining a simple autoencoder, and then discards the decoder part. Afterwards, it fine-tunes the parameters of the encoder with a novel loss function based on the *invariance of distribution* property. The invariance of distribution is based on the assumption that the conditional distribution of any data point given the cluster centers is invariant to different distance metrics. The conditional distribution for each data point is a $c$-dimensional vector providing a probability that is related to the closeness between the given data point and the centroids of the $c$ clusters, so that the data point has a higher probability to be assigned to a closer cluster. The closeness is measured by different metrics such as the Euclidean and cosine similarity metrics. The invariance of distribution uses the KL-divergence to minimize the discrepancy among different distributions defined by different closeness measures. After updating the encoder parameters and the cluster centroids in the fine-tuning step, each data point is assigned to the cluster with the closest centroid. Nevertheless, it is not clear how minimizing the discrepancy among different distributions benefits the SC or affects the geometric structure of the data from multiple manifolds. Moreover, CSC only uses the autoencoder network for the initialization of the parameters of the encoder.

The idea of explicitly merging self-expressiveness in autoencoder networks for SC was first brought to light in the framework of Deep Subspace Clustering network (DSC-Net) [42]. DSC-Net is the pioneer work to identify that the linear combination in the collaborative representation corresponds to a layer of fully connected neurons without non-linear activations and biases. This layer, dubbed the *self-expressive layer*, is added between the encoder and decoder of a standard autoencoder. The parameters (weights) of this layer can be interpreted as the coefficient matrix $C$. The authors promoted the use of convolutional autoencoders as the backbone architecture, reasoning that they can be trained easier compared to the classical fully connected autoencoders as they have less parameters. This has been recently proved theoretically in [57]. Let $\Theta$ denote the parameters of the encoder and the decoder. The self-expressiveness in the latent space, that is, $Z_\Theta \approx Z_\Theta C$, is a linear layer located between the encoder and decoder, as illustrated in Figure 5. In particular, all the input data $X$, as a single large batch, is fed into the encoder to obtain the embedded data $Z_\Theta$, where $Z_\Theta(:,i)$ corresponds to the data point $X(:,i)$. The self-expressive representation of the latent data $Z_\Theta$ is calculated using the subsequent linear layer. The decoder reconstructs each data point in the input batch from the columns of $Z_\Theta C$. In order to seek a tradeoff between the reconstruction and self-expression errors, DSC-Net solves the following optimization problem:

$$\begin{aligned} \min_{\Theta, C} \; & \frac{1}{2}||X - \tilde{X}_\Theta||_F^2 + \frac{\lambda_1}{2}||Z_\Theta - Z_\Theta C||_F^2 + \lambda_2 ||C||_p^p, \\ & \text{such that } \operatorname{diag}(C) = 0, \end{aligned} \tag{22}$$

where $||\cdot||_p$ is the component-wise $\ell_p$ norm that regularizes the coefficient matrix $C$ (that is, the weights of the fully connected layer). Two different regularizations were considered in [42], namely $p = 1, 2$, leading to DSC-Net-L1 and DSC-Net-L2 networks, respectively. The network is trained in two steps: (i) pretraining which includes initialization of the autoencoder without the self-expressive layer, and (ii) fine-tuning which includes training the whole network including the self-expressive layer. Once the network is trained, the weights $C$ of the self-expressive layer are clustered using spectral clustering, as in linear SC.

**Recent extensions of self-expressive autoencoders** DSC-Net initiated many extensions that aim to integrate the representational power of neural networks with self-expressive SC algorithms. Among noticeable extensions, there are two approaches that attempt to unify the self-expressive autoencoders with the concept of self-supervision/self-training. The Self-Supervised Convolutional

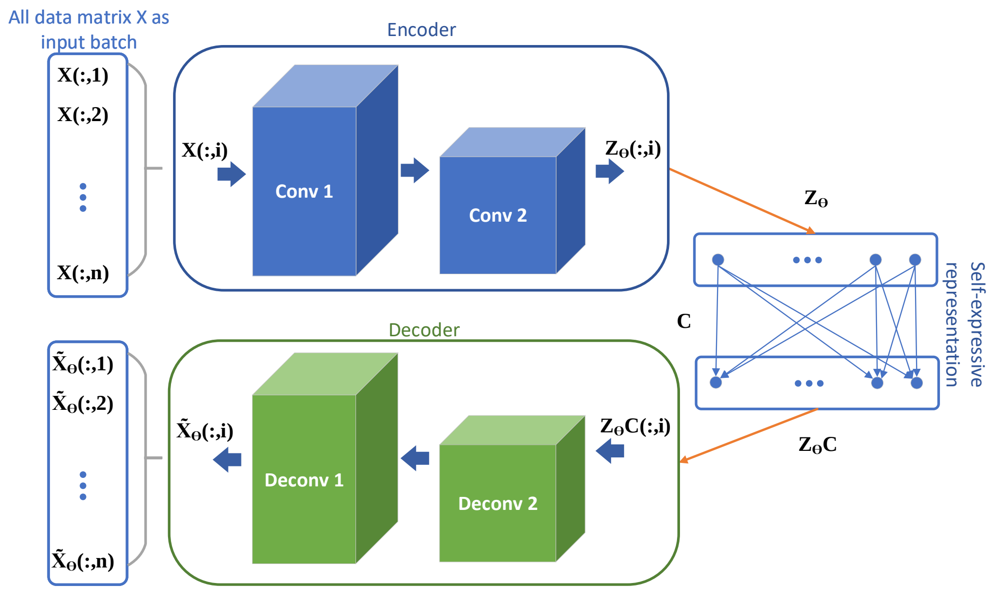


Figure 5: Illustration of autoencoder based subspace clustering using fully connected linear self-expressive layer in DSC-Net [42]. Two layers of convolutional encoder and two layers of deconvolutional layers are shown for the decoder. All the input data are used as a large single batch for minimizing the loss function and training the autoencoder.

Subspace Clustering Network (S$^2$ConvSCN) [140] combines DSC-Net with a classification convolutional module (for feature learning) and a spectral clustering module (for self-supervision) into a joint framework. This framework uses the (noisy) clustering labels of the spectral clustering module to supervise the training of the feature learning network and the self-expressive coefficient learning. The idea of collaborative learning between the classification module and the self-expressive layer was reused in [143], but using the concept of confidence learning. In contrast to S$^2$ConvSCN, the model in [143] supervises the training without the use of the computationally expensive spectral clustering. In particular, two affinity matrices are constructed to supervise the training by selecting highly confident sample pairs: (i) the affinity matrix of a *binary* classifier module based on a convolutional network, which indicates whether two data points belong to the same cluster or not, and (ii) the affinity matrix corresponding to the weights of the self-expressiveness layer. Nonetheless, selecting confident samples depends on thresholding parameters which are hard to tune in an unsupervised setting.

Other notable extensions include the following. Sparse and low-rank regularized DSC (SLR-DSC) [154] regularizes the coefficient matrix using the nuclear and Frobenius norms. An adaptation of DSC-Net for multi-modal data was presented in [1]. Zhou et al. [151] added a distribution consistent loss term to keep the consistency between two distributions of the original data and the embedded representation (the output of the encoder). Inspired by human cognition, Jiang et al. [45] aggregated a self-paced learning framework [44] with the DSC-Net loss function to encourage the network to learn “easier” samples at first. Easier samples are defined as the ones with a small value of the reconstruction and of the self-expressive representation errors. However, similar to many self-paced frameworks, it is very sensitive to thresholding parameters [27]. Instead of focusing on the learned features from the output of the encoder, Kheirandish et al. [53] proposed a multi-level representation of DSC-Net (MLRDSC) which combines low-level information from the initial layers with high-level information from the deeper layers.

### 3.3.3 Deep adversarial subspace clustering

So far we have reviewed deep SC approaches that are based on a discriminative supervised neural network (such as those for self-supervised feature learning) and autoencoder architectures. There are a few approaches that rely on *Generative Adversarial Networks* (GANs) [26]. GANs are composed of two modules, a *generator* and a *discriminator*. The generator module produces new "fake" samples by learning to map from a latent space (often based on random uniform or Gaussian distributions) to the unknown desired distribution of the samples from the given data set. The goal of the generator is to produce samples that follow the true distribution of the data points as closely as possible. On the other hand, the goal of the discriminator is to distinguish the generated fake samples from the "real" samples of the data set. These two modules are trained through a minimax game such that the success of each module is the loss of the other one. GANs are often difficult to train and this explains the fewer number of works that combines GANs with SC.

The first SC algorithm that adopted the GAN architecture is Deep Adversarial Subspace Clustering (DASC) [152]. In order to match the goal of SC, DASC modifies the generator and discriminator modules based on well-known linear subspace properties:

- DASC generator. The generator has two main components: a self-expressive autoencoder (similar to DSC-Net, with the encoder and decoder network parameters denoted by $\Theta$) and a sampling layer for producing fake data for each estimated subspace. The generator first uses a deep self-expressive auto-encoder to transform the data $X$ into the latent variable $Z_\Theta$ such that they are encouraged to be located in a union of linear subspaces. Thereafter, spectral clustering is applied on the weights from the self-expressive layer ($C$) to obtain the clustering assignment. The generator produces new "fake" samples $\{\bar{Z}_\Theta^{(i)}\}_{i=1}^c$ by sampling from the estimated subspaces using property of linear subspaces: *linearly combining samples within a subspace generates a sample from the same subspace.* To this end, the fake samples are generated by random linear combination of the samples within each cluster (the weights of the combinations are chosen uniformly at random in $[0, 1]$). Apart from the fake data generation process, the definition of the "real" data is also noticeably different from the classic GAN for which the real data is the given data set. In DASC, the real data $\{\tilde{Z}_\Theta^{(i)}\}_{i=1}^c$ are a predefined fraction of the samples within each estimated cluster that have a small projection residual onto the learned subspaces by the discriminator. Mathematically, the generator solves

$$\min_{C,\Theta} \quad \underbrace{\frac{1}{c}\sum_{i=1}^{c}\frac{1}{n_i}\sum_{j=1}^{n_i} ||\bar{Z}_\Theta^{(i)}(:,j) - U_iU_i^\top \bar{Z}_\Theta^{(i)}(:,j)||_2^2}_{\text{sum of projection residual of fake data from each subspace}} \tag{23}$$
$$+ \quad \underbrace{\lambda_1||X - \tilde{X}_\Theta||_F^2}_{\text{reconstruction loss of autoencoder}} \quad + \quad \underbrace{\lambda_2||Z_\Theta - Z_\Theta C||_F^2 + \lambda_3||C||_F^2}_{\text{LSR based self-expressiveness in embedded space}},$$

where $\Theta$ and $\tilde{X}_\Theta$ denote the parameters of the network and the reconstructed data in the autoencoder, respectively, and $n_i$ is the number of fake samples for the $i$th subspace/cluster. The matrices $U_i$ ($i = 1, \ldots, c$) are the learned bases for each subspace in the discriminator.

In fact, the first term in (23) is the sum of projection residuals of the fake data for all of the estimated subspaces. The generator tries to "fool" the discriminator by producing (embedded) samples that are very close to the estimated subspaces by the discriminator.

- DASC discriminator. The discriminator is parametrized by the basis vectors of each subspace of the nonlinear transformed data ($Z$). The basis vectors of each subspace/cluster are learned such that the projection residual loss function for the real data is smaller compared to the fake ones. The discriminator solves

$$\min_{\{U_i\}_{i=1}^c} \frac{1}{c}\sum_{i=1}^{c}\frac{1}{n_i}\sum_{j=1}^{n_i}\Big(\underbrace{||\tilde{Z}_\Theta^{(i)}(:,j) - U_iU_i^\top \tilde{Z}_\Theta^{(i)}(:,j)||_2^2}_{\text{projection residual of } \textit{real} \text{ data}} + \big[\epsilon - \underbrace{||\bar{Z}_\Theta^{(i)}(:,j) - U_iU_i^\top \bar{Z}_\Theta^{(i)}(:,j)||_2^2}_{\text{projection residual of } \textit{fake} \text{ data}}\big]_+\Big)$$

$$+ \underbrace{\lambda_1\sum_{i=1}^{c}||U_i^\top U_i - I||_F^2}_{\text{approximately orthonormal bases}} + \underbrace{\lambda_2\sum_{i\neq j}||U_i^\top U_j||_F^2}_{\text{separibility of subspaces}}, \tag{24}$$

where $\big[x\big]_+ = \max(x,0)$. The first term in (24) ensures that the discriminator learns the bases such that they fit the intrinsic subspace of each cluster for the real data. The second term is the adversarial goal of the discriminator compared to the generator. This term ensures that the learned bases of the discriminator produce sufficiently large residuals for the generated fake data.

Subsequently, the generator and discriminator are trained such that the generator is encouraged to produce fake data close to the subspaces learned by the discriminator which leads to higher clustering quality.

However, DASC completely ignores the distribution of the input data $X$ and merely focuses on the latent embedded data $Z$. Recently, Yu et al. [137] proposed two extensions to DASC. In the first approach, an additional adversarial learning is utilized to model the distributions of the input data along with the adversarial learning for the corresponding latent representations (similar to DASC). In the second approach, inspired by the self-supervised SC approach in [143], the adversarial learning in the latent space is replaced by a self-supervised module to encourage the (autoencoder) network to learn discriminative embedding (features) for each subspace.

**Remark 4.** *Instead of using GANs to enhance SC, one could also use linear SC to overcome challenges of classical GANs, such as the "mode-collapse" issue [5], that is, the lack of sample variety in the generator's output. For example, Liang et al. [58] suggested to use a clustering module based on linear SC to exploit subspaces within the latent representation of the data. The clustering output is used by the generator to produce samples conditioned on their corresponding subspace to promote diverse sample generation from all latent subspaces.*

## 4 Computational cost of nonlinear SC algorithms

Despite the nice theoretical properties of self-expressive representations [95, 21] and their practical efficiency, expressing the data points using other data points from the data set can be computationally inefficient. In fact, the computational complexity for computing the coefficient matrix in the major linear SC algorithms is either quadratic or cubic in the number of data points [86], and hence using them for medium and large scale data sets is computationally prohibitive. To address the scalability issue, several approaches were proposed which either limit the self-expression over a smaller size dictionary rather than the whole data [2, 101, 11, 133] or propose specific solvers (such as the greedy OMP [135] or using proximal gradient descent framework [86]).

**Remark 5.** *In addition to the high computational complexity of calculating the self-expressive representation (the first step of most nonlinear SC algorithms), the spectral clustering step has in general a computational cost of $O(n^3)$ as well [38]. However, structures in the affinity matrix such as sparsity can reduce the computational time significantly [11]. Moreover, there exists fast approximation algorithms for spectral clustering [123, 102, 38]. Hence, the main focus of scalable extensions of linear SC was to provide faster and more efficient approaches to (approximately) compute the coefficient matrix.*

The computational burden of self-expressive representations is present in the majority of nonlinear extensions reviewed in this survey as well. Representing data points using the whole data set, whether it is in the ambient or in the (implicit) embedding space, typically has the computational cost of $O(n^3)$ or $O(dn^2)$. Similar to linear SC, ADMM is the standard algorithm used for the majority of the optimization problems. The major bottleneck is computing the $n \times n$ coefficient matrix which often involves solving an $n$-by-$n$ linear system for Kernel based and locality preserving approaches (in the subcategories of tangent space approximation and avoiding cannot-links). Solving this problem in each iteration of the ADMM framework has a computational complexity of $O(n^3)$, or $O(dn^2)$ using the matrix inversion lemma (also known as Sherman-Morrison-Woodbury identity [35], see [86, Remark 1] for more details). Using the graph Laplacian regularizer, an additional bottleneck of computing a pairwise similarity matrix in the ambient space is added (in $O(dn^2)$ operations), and, moreover, the coefficient matrix should be computed by solving a Sylvester equation with the complexity of $O(n^3)$. The high computational cost is also a very evident drawback of the self-expressive based autoencoders in neural network approaches. In fact, the number of parameters of the fully-connected self-expressive layer is $\mathcal{O}(n^2)$. Computing the latent representation of the encoder has the time complexity of $O(d_e n^2)$ where $d_e$ is the dimension of the encoder output.

In addition to the time complexity, the memory requirement is often significant as well. Storing (and clustering) an $n \times n$ coefficient matrix is usually restrictive, unless the coefficient matrix is sparse. Computing and storing $n \times n$ (dense) Gram matrices increases the space complexity of kernel based approaches further. This is even worse for MKL algorithms which update the kernel matrix at each iteration of an ADMM based algorithm.

**Scalable nonlinear SC approaches** In contrast to linear SC, the scalability issue of nonlinear alternatives has not yet been investigated much. To the best of our knowledge, the only scalable algorithm in the category of locality preserving SC is an avoiding cannot-link approach [31] which uses $k$ nearby data points to represent each data point, with a computational cost of $O(k^2nd)$ operations.

With neural networks based SC approaches gaining increasing attention, some approaches tried to tackle the computational bottleneck of DSC-Net, which is due to the self-expressive layer. Zhang et al. [142] suggested to replace this layer with a more computationally effective module. Specifically, instead of directly pursuing self-expressive representation in the latent space and constructing an affinity matrix, an iterative linear SC algorithm, namely, the k-subspace clustering (k-SC) [36] was revisited. The k-SC approach alternates between assigning the data to individual subspaces and estimating the parameters of each subspace. However, as discussed in Section 1, a significant disadvantage of iterative linear SC approaches, including k-SC, is that they require the dimension of each subspace to be known in advance. Instead of changing the architecture, Seo et al. [93] proposed an efficient optimization framework for DSC-Net which uses a closed-form solution for deriving the weights of the self-expressive layer using Lagrangian multipliers. However, not only the obtained accuracy is lower than DSC-Net, but also the computational complexity is still quadratic in the number of samples.

## 5 Evaluation and numerical comparison

In this section, we investigate the properties and performances of major nonlinear SC approaches on synthetic and real world data sets.

**Selected Algorithms** Among many nonlinear SC algorithms covered in this survey, eight major methods are considered for a detailed comparison and evaluation. The selected algorithms within each category are summarized in Table 2 and described as follows:

Table 2: Selected representative methods for numerical comparison

| **Category** | **subcategory** | **method** | **characteristic** | **year** |
|---|---|---|---|---|
| Locality preserving | graph laplacian regularization | SMR [37] | Graph Laplacian regularized self-expressive representation | 2014 |
| | | LR$\ell_1$-SSC [129] | sparse Graph Laplacian regularization | 2014 |
| | avoiding cannot-links | KNN-SSC [155] | using local self-expressive dictionary | 2016 |
| | tangent space approximation | SMCE [20] | tangent approximation using adaptive neighborhood size | 2011 |
| Kernel-based | single kernel | KSSC [79] | kernelized SSC | 2014 |
| | multiple kernels | LKG [51] | MKL with low-rank consensus kernel | 2019 |
| Neural-network based | self-expressive latent representation learning | DSC-Net [42] | autoencoder with self-expressive layer | 2017 |
| | deep adversarial subspace clustering | DASC [152] | combining adversarial learning with LSR | 2018 |

- *Locality preserving based approaches*:
  - **Smooth Representation (SMR)** [37] is in the category of locality preserving SC based on Laplacian regularization. The laplacian matrix in SMR is obtained from a binary K-NN similarity matrix (with parameter $k$).
  - **Laplacian Regularized $\ell_1$-SSC (LR$\ell_1$-SSC)** [129] is similar to SMR but with additional sparse regularization. For a fair comparison, we use the same Laplacian matrix $L$ as SMR.[6,7]

[6] In [129], Gaussian kernels are used for initializing the Laplacian matrix $L$. Then, at each iteration of the optimization process, the Laplacian matrix $L$ is updated using the coefficient matrix $C$ from the previous iteration. For a fair comparison, we use the same fixed Laplacian matrix based on binary K-NN, similar to SMR. Moreover, convergence of such a scheme is not guaranteed.

[7] Optimizing the Laplacian regularized SC problem with additional non-differentiable regularizations (such as the $\ell_1$ or nuclear norm) using ADMM involves solving Sylvester equations [130]. Some approaches in the literature mistakenly

- **k-nearest neighbors based sparse subspace clustering (KNN-SSC)** [155] is the category of avoiding cannot-links. This approach uses only data points in the k-nearest neighbors of samples as the dictionary for the self-expressive representation. We use the efficient scalable implementation provided in [98].
- **Sparse Manifold Clustering and Embedding (SMCE)** [20] falls in the category of Tangent space approximation. It is one of the early nonlinear extensions of SSC (by the same authors) and the most notable tangent space nonlinear SC approach.

- *Kernel based subspace clustering*:

  - **Kernel Sparse Subspace Clustering (KSSC)** [79] is the pioneer single kernel based nonlinear SC approach based on sparsity regularization. We consider Gaussian (RBF) kernel for the experiments. Note that KSSC with a linear kernel is equivalent to classical linear SSC algorithm [79].
  - **Low-rank Kernel Learning for Graph matrix (LKG)** [51] is a multiple kernels based nonlinear SC approach. LKG learns a low-rank kernel from a group of predefined kernels, solving the optimization problem (18); see Section 3.2.2. We follow the generic setting introduced in [50] and consider 12 kernels: 7 Gaussian kernels defined, for $i = 1, 2, \dots, 7$, as
  $$K_G^{(i)}(x, y) = \exp^{\frac{-||x-y||_2^2}{t_i d_{\max}^2}}, \quad t_i \in \{0.01, 0.05, 0.1, 0.5, 1, 10, 50\}$$
  where $d_{\max}$ is the maximal distance between two data points, a linear kernel $K_G^{(8)}(x, y) = x^\top y$, and 4 polynomial kernels of the form $\left(a + x^\top y\right)^b$ for all possible values of $a \in \{0, 1\}$ and $b \in \{2, 3\}$. Following the procedure in [50], all kernel matrices are normalized to have values between 0 and 1.

- *Neural networks based subspace clustering*:

  - **Deep Subspace Clustering Network (DSC-Net)** [42] is arguably the most notable neural-network based nonlinear SC algorithm which introduced the self-expressive fully connected layer in the autoencoder architecture. DSC-Net can be used with $\ell_1$ and $\ell_2$ regularizations for the columns of the coefficient matrix, leading to DSC-Net-L1 and DSC-Net-L2 algorithms, respectively. For DSC-Net, we use both the convolutional autoencoder version provided by the authors and a modified *fully connected autoencoder* version (with tanh activation function) implemented by ourselves. The fully connected autoencoder is used for synthetic data sets where the ambient dimension is small.
  - **Deep Adversarial Subspace Clustering (DASC)** [152] is the main existing approach in the literature for the combination of self-expressive linear SC with GANs; see Section 3.3.3.

Moreover, three linear SC algorithms are used as baselines: SSC, with both Frobenius norm and $\ell_1$ norm regularization for the error matrix (denoted by SSC-L2 and SSC-L1, respectively), and LRR with $\ell_1$ regularization for the reconstruction error.

derive linear systems instead of Sylvester equations (this error comes from the derivative of $\text{tr}(CLC^\top)$, authors use $2LC$ instead of $2CL$). Hence, we use our own implementation for the experiments.

The code for SMR[8], SMCE[9], KNN-SSC[10], DSC-Net[11] and DASC[12] are available from the author's website. The rest of the algorithms, implemented by us, are available from `https://sites.google.com/site/nicolasgillis/code`. The code contains the selected parameters for the algorithms for each data set as well. The parameters of each algorithm are chosen following the recommendations of the authors.

**Quality Measures** From the many possible metrics to quantify the performances of (subspace) clustering algorithms, two main metrics are commonly used to evaluate the clustering performance: the Accuracy (ACC) and the Normalized Mutual Information (NMI) [97]. Given the ground-truth labels (clusters) $\ell \in \{1, \ldots, c\}^n$ and the obtained labels $\hat{\ell} \in \{1, \ldots, c\}^n$ by a clustering algorithm, where $c$ is the number of clusters, these metrics are defined as follows:

$$\text{ACC} = \max_{\pi, \text{ a permutation}} \frac{\sum_{i=1}^{n} \mathbf{1}\big(\ell(i) = \hat{\ell}_\pi(i)\big)}{n}, \quad \text{and} \quad \text{NMI} = \frac{I(\ell; \hat{\ell})}{\sqrt{H(\ell)H(\hat{\ell})}},$$

where $\mathbf{1}(\cdot)$ is the indicator function which returns one if the input condition is correct, $\hat{\ell}_\pi$ is the labelling obtained after permuting the $c$ labels using $\pi$, $I(\cdot;\cdot)$ is the mutual information metric, and $H(\cdot)$ is the entropy function. The values of ACC and NMI belong to the interval $[0, 1]$; a higher value indicates a better clustering performance.

## 5.1 Synthetic data

The performances of the nonlinear SC approaches are compared on both linear and nonlinear synthetic data sets. We believe that analyzing the properties of nonlinear approaches on carefully designed linear data sets is helpful in understanding the characteristic of these approaches. In fact, linear subspaces are special cases of nonlinear ones, and hence a nonlinear approach should work in linear scenarios. For a fair comparison of the algorithms, no post-processing is performed on the coefficient matrices. The obtained coefficient matrices are only symmetrized and then spectral clustering is applied.

**Parameters** For locality preserving approaches, namely, SMR, KNN-SSC and LR$\ell_1$-SSC, the parameter $k$ which indicates the number of nearest neighbors in the similarity matrix construction is set to 10. For the linear synthetic datasets, we did not notice sensitivity to this parameter as long as it is not below 10 (but as this value increases the computational time increases as well). The sensitivity to the parameter $k$ is more noticeable for the nonlinear datasets. We obtained similar results for $k \in [10, 15]$, while larger values for $k$ leads to increasingly wrong connections. For KSSC, we tuned the parameter $\sigma$ of the RBF kernel from the set {0.01,0.05,0.1,0.5,1,2,5,10,20} for the best overall accuracy and set it as 1 for linear synthetic datasets (we obtained similar result for $\sigma = \{0.5, 1, 2\}$), and as 0.05 for the nonlinear datasets. The nonlinear datasets are more sensitive to the value of $\sigma$ which is due to the complex nonlinear structures in these datasets. In general KSSC shows significant sensitivity to the parameters which is a common challenge of kernel based approaches. Moreover, the proper value of $\sigma$ depends on the regularization parameter which balances the trade-off between

[8]`http://zero-lab-pku.github.io/publication/limingjie/cvpr14_smooth_representation_clustering/`
[9]`http://khoury.neu.edu/home/eelhami/codes.htm`
[10]`http://github.com/sjtrny/kssc`
[11]`http://github.com/panji530/Deep-subspace-clustering-networks/`
[12]`http://github.com/hyqneuron/dsc_gan`

sparsity and self-expressiveness. Sparser solutions are more sensitive to lower values of $\sigma$ which might lead to over-segmentation. In contrast, denser solutions are more sensitive to larger values of $\sigma$ which are prone to dense wrong connections. For DSC-Net, we consider three hidden layers for the encoder with {10,8,4} units for the linear synthetic data sets (we obtained similar results as long as no layer has less than 4 neurons), and {8,4,2} units for the nonlinear synthetic data sets. Increasing or decreasing the number of hidden layers did not affect the results significantly. The encoder in DASC has two hidden layers with {8,4} and {2,2} units for the synthetic linear and nonlinear data sets, respectively. For a detailed report on the selected parameters for each approach, see Appendix B.

### 5.1.1 Linear subspace clustering

Two principal arrangements of subspaces are considered for the generation of linear synthetic data sets: independent and disjoint. Intuitively, and theoretically [95], the separation between the subspaces plays a critical role in recovery of the subspaces. Hence, for the two types of synthetic data sets, we consider a semi-random data generation approach where the bases of the subspaces are carefully determined with a parameter that controls the affinity between the subspaces. The data generation model, which is inspired by the models in [95, 2], is explained in details in the Appendix A. In this section, we compare the performance of nonlinear approaches for both synthetic independent and disjoint subspaces.

**First experiment: two independent subspaces** Two linearly independent subspaces with intrinsic dimension two are first considered. We set $N = 900$ (with 450 points per subspace), $d_i = 2$ for $i = 1, 2$, and $d = 20$ (the ambient dimension is 20). The angle between the two subspaces is controlled by the parameter $\theta \in \{5, 10, 20, 30, 45, 60, 90\}$. For each value of $\theta$, 10 sets of data points are randomly generated to analyze the performance of the tested algorithms. Table 3 reports the average and standard deviation of the accuracy depending on the different values of $\theta$.

We observe the following:

- For a sufficiently small affinity between subspaces (more precisely, for $\theta \geq 45$), all of the approaches provide an almost perfect clustering (namely, the accuracy is above 98.97% in all cases).
- As guaranteed by the theory [79, 21, 37], SMR, SSC-L2 and LRR perform almost perfectly for all cases of independent subspaces, and their performance is not affected by the affinity between the subspaces. The performance of SSC-L1 was identical to SSC-L2. This is expected as the data set is not contaminated by noise so that the data fitting term does not affect the result.
- Interestingly, LR$\ell_1$-SSC, which integrates Laplacian regularization via the $\ell_1$ norm, also leads to subspace preserving coefficients for all values of $\theta$ although this is not (yet) theoretically supported.
- KNN-SSC, which prevents connections between faraway data points, performs poorly for close subspaces (namely, $\theta = 5, 10$). However, we noticed that by increasing the number of nearest numbers to more than 100, KNN-SSC leads to an almost perfect clustering result.
- SMCE and KSSC with RBF kernel are producing subspace preserving coefficients for $\theta$ sufficiently large. This is expected, because SMCE relies on the proximity of data points for the tangent space approximation, and KSSC with RBF kernel is dependent on the nearby samples

Table 3: The average and standard deviation of the accuracy, in percent, of SC approaches on the two independent linear subspaces separated by an angle $\theta$.

| | $\theta =$ | 5 | 10 | 20 | 30 | 45 | 60 | 90 |
|---|---|---|---|---|---|---|---|---|
| SMCE | mean | 51.17 | 50.56 | 98.45 | 99.72 | 99.94 | 99.94 | 99.95 |
| | std | 1.14 | 0.52 | 0.50 | 0.25 | 0.07 | 0.05 | 0.05 |
| SMR | mean | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | std | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| KSSC(RBF) | mean | 51.25 | 51.85 | 99.84 | 99.96 | 99.97 | 99.97 | 100 |
| | std | 0.71 | 0.99 | 0.13 | 0.05 | 0.04 | 0.04 | 0 |
| LR$\ell_1$-SSC | mean | 99.88 | 99.97 | 100 | 100 | 100 | 100 | 99.98 |
| | std | 0.09 | 0.04 | 0 | 0 | 0 | 0 | 0.03 |
| LKG | mean | 51.47 | 51.42 | 51.13 | 50.72 | 99.18 | 99.63 | 99.67 |
| | std | 0.72 | 1.34 | 0.81 | 0.73 | 0.65 | 0.26 | 0.26 |
| KNN-SSC | mean | 51.60 | 51.44 | 98.26 | 99.35 | 99.85 | 99.86 | 99.82 |
| | std | 0.97 | 0.81 | 0.63 | 0.38 | 0.16 | 0.08 | 0.17 |
| DSC-Net-L2 | mean | 51.41 | 60.65 | 94.14 | 97.56 | 99.44 | 99.46 | 99.33 |
| | std | 0.96 | 14.09 | 5.37 | 1.98 | 0.45 | 0.52 | 0.49 |
| DSC-Net-L1 | mean | 51.33 | 50.84 | 83.82 | 96.11 | 98.97 | 99.51 | 99.57 |
| | std | 1.06 | 0.91 | 8.68 | 6.03 | 0.76 | 0.25 | 0.34 |
| DASC | mean | 51.70 | 52.22 | 94.15 | 98.95 | 99.93 | 99.85 | 99.91 |
| | std | 1.19 | 1.40 | 5.62 | 2.57 | 0.09 | 0.16 | 0.12 |
| SSC-L2 | mean | 99.92 | 99.91 | 99.92 | 99.96 | 99.96 | 99.95 | 99.94 |
| | std | 0.07 | 0.08 | 0.07 | 0.03 | 0.03 | 0.04 | 0.05 |
| LRR | mean | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | std | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

which is controlled by the parameter $\sigma$. Hence, the closer the subspaces get, the higher the chance of wrong clusterings.

- The accuracy of DSC-Net increases with $\theta$. Interestingly, DSC-Net based on $\ell_1$ regularization perform worse than with $\ell_2$. The authors in [42] argued that this happens in practice because $\ell_1$ is non-differentiable at zero. However, we do believe this is related to optimizing the network weights (including the coefficient matrix) using the default subgradient descent method of neural networks. In practice, they usually produce non-sparse solutions compared to other optimization methods such as proximal methods [3]. Generating non-sparse solutions is due to slow convergence. Although spectral clustering is expected to be robust to weak wrong connections, there is no guarantee that it is robust to the intermediate non-sparse solutions; in particular if the solution contains many small non-zero entries with no dominant large entries.

- LKG has the worst performance, providing high accuracy only for $\theta \geq 45$. This shows that multiple kernel learning might lead to worse performance compared to the single kernel based alternatives.

**Second experiment: three disjoint subspaces** In the second noiseless experiment, samples are generated from three linearly disjoint subspaces. In particular, we set $N = 900$ (with 300 data points per subspace), $d_i = 2$ for $i = 1, 2, 3$, $d = 20$ and $\theta \in \{5, 10, 20, 30, 45, 60\}$ (note that we do not consider $\theta = 90^\circ$ because it leads to identical subspaces for the first and second basis); see Appendix A for more details on the data generation model. The average and standard deviation of accuracy over 10 trials for 10 randomly generated subspaces are reported in Table 4.

We observe the following:

- Compared to the independent case, the accuracy of all nonlinear SC approaches decreases significantly for $\theta$ sufficiently small (high affinity between subspaces). However, sparsity based approaches, namely, SMCE, KSSC, LR$\ell_1$-SSC, KNN-SSC and DSC-Net-L1 have the best performances, especially for $\theta$ sufficiently large. This is inline with theoretical justifications for sparse based linear SC algorithms, that is, SSC [95] and $\ell_0$-SSC [128].

- As opposed to the first experiment with independent subspaces, the performance of SMR is among the worse for disjoint subspaces. This is expected because SMR only considers the *grouping effect* with no other regularization. Similar performance is observed for DSC-Net-L2 and DASC which are based on $\ell_2$ regularization. This highlights the vital role of sparsity in dealing with disjoint subspaces.

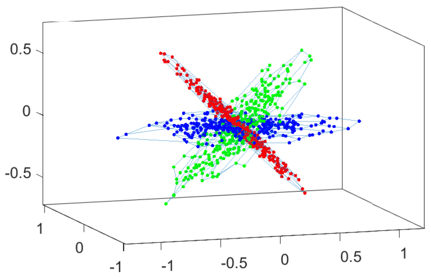
- Interestingly, KSSC with RBF kernel, KNN-SSC and SMCE perform slightly better than SSC-L2 and LR$\ell_1$-SSC for $\theta = 30^\circ$. This suggests that nonlinearity might lead to better representation learning when subspaces are close (but with sufficient separability) as shown empirically in [79] for KSSC.

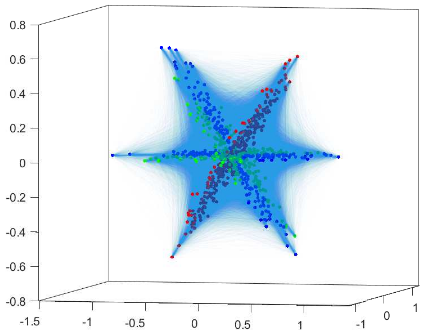
Figure 6 displays the connectivity graphs of the three disjoint subspaces for $\theta = 45^\circ$, for the 9 tested algorithms. The data points for three clusters are shown in red, blue and green. The first three dimensions are considered for these plots. Note the dense and many wrong connections for SMR, DSC-Net-L2 and DASC. There are many wrong weak connections in the graph of DSC-Net-L1 too,

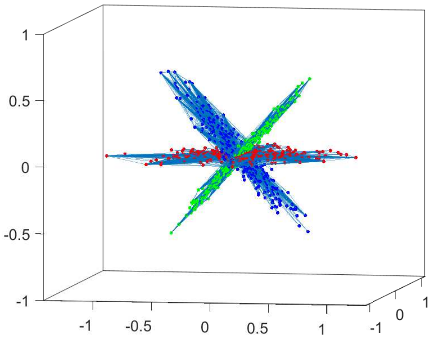
Table 4: The average and standard deviation of the accuracy, in percent, of SC algorithms on three disjoint linear subspaces whose affinity is measured by the angle $\theta$.

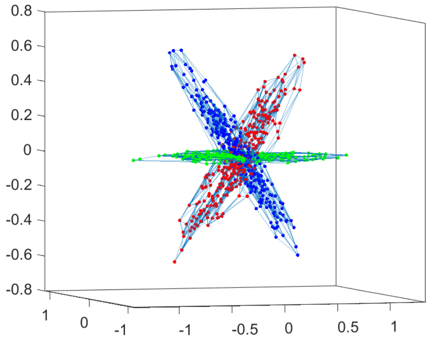
| | $\theta =$ | 5 | 10 | 20 | 30 | 45 | 60 |
|---|---|---|---|---|---|---|---|
| SMCE | mean | 35.53 | 35.54 | 63.81 | 99.25 | 99.73 | 99.92 |
| | std | 0.56 | 0.74 | 9.73 | 0.35 | 0.17 | 0.07 |
| SMR | mean | 67.61 | 67.22 | 66.97 | 66.93 | 67.70 | 35.70 |
| | std | 0.54 | 2.96 | 2.81 | 3.10 | 1.11 | 1.02 |
| KSSC(RBF) | mean | 35.42 | 35.74 | 54.82 | 99.47 | 99.78 | 99.95 |
| | std | 0.57 | 0.93 | 10.18 | 0.28 | 0.20 | 0.07 |
| LR$\ell_1$-SSC | mean | 62.93 | 80.72 | 84.76 | 92.94 | 99.96 | 99.98 |
| | std | 11.65 | 11.06 | 13.49 | 11.51 | 0.05 | 0 |
| LKG | mean | 35.72 | 35.30 | 35.86 | 37.23 | 97.50 | 99.03 |
| | std | 0.94 | 0.70 | 1.02 | 1.52 | 0.90 | 0.68 |
| KNN-SSC | mean | 35.74 | 35.86 | 42.13 | 98.85 | 99.72 | 99.76 |
| | std | 0.71 | 1.15 | 7.02 | 0.59 | 0.13 | 0.09 |
| DSC-Net-L2 | mean | 39.50 | 54.58 | 62.88 | 64.73 | 60.93 | 38.35 |
| | std | 6.60 | 6.05 | 3.62 | 1.85 | 4.65 | 2.25 |
| DSC-Net-L1 | mean | 35.18 | 40.46 | 57.93 | 61.93 | 97.91 | 98.55 |
| | std | 0.72 | 5.53 | 6.02 | 7.07 | 1.34 | 0.82 |
| DASC | mean | 35.94 | 59.42 | 65.45 | 65.61 | 67.44 | 37.03 |
| | std | 0.87 | 5.06 | 4.28 | 3.91 | 1.20 | 1.45 |
| SSC-L2 | mean | 60.95 | 83.15 | 82.77 | 92.84 | 99.76 | 99.84 |
| | std | 11.12 | 10.40 | 13.94 | 11.41 | 0.18 | 0.13 |
| LRR | mean | 67.63 | 67.22 | 66.97 | 66.92 | 67.70 | 35.75 |
| | std | 0.55 | 2.96 | 2.81 | 3.09 | 1.11 | 1.07 |

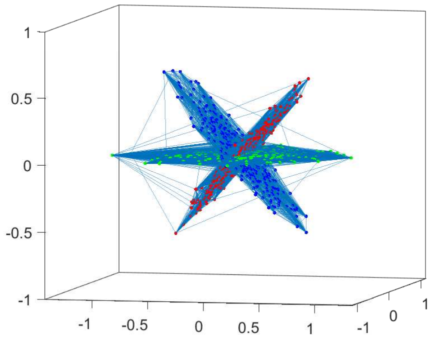
however, the stronger connections within subspaces overcome the weaker wrong ones, leading to an almost correct clustering result (namely, 97.91%; see Table 4). Non-sparse solution of DSC-Net-L1 is expected as subgradient descent based methods usually do not lead to sparse solutions in practice [3].

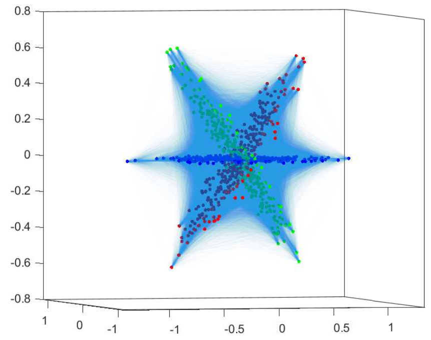

(a) SMCE (99.78%)

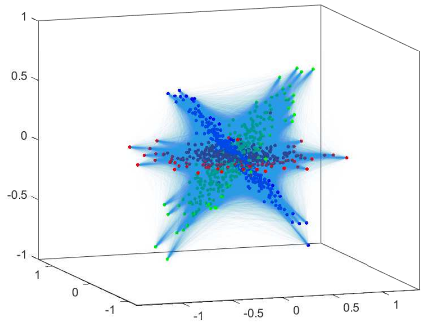

(b) SMR (66.67%)

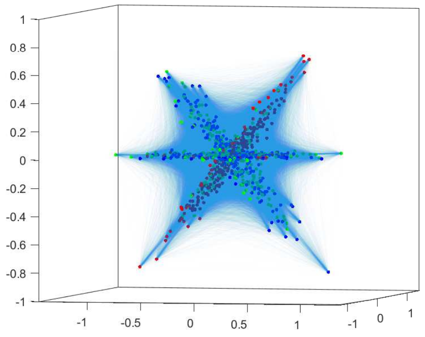

(c) LR$\ell_1$-SSC (99.89%)


(d) KNN-SSC (99.78%)


(e) KSSC (RBF kernel) (100%)


(f) LKG (99.22%)


(g) DSC-Net $\ell_1$ (99.56%)


(h) DSC-Net $\ell_2$ (60.89%)

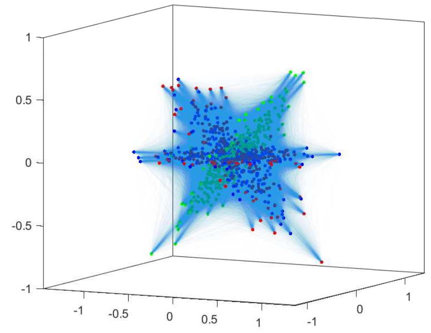

(i) DASC (66.70%)

Figure 6: The connectivity graphs of nonlinear SC algorithms on three disjoint subspaces with $\theta = 45°$. The accuracy of the displayed instances is indicated in brackets. The average accuracy over the 10 randomly generated instances can be found in Table 4.

**Conclusion for the linear subspace clustering experiments** The majority of nonlinear SC approaches have difficulties in segmenting the subspaces that are close to each other. This is specifically evident for the cases with $\theta = 5, 10$. This is expected since the majority of non-linear approaches have the implicit assumption that the data points from different clusters are not spatially close to each other. This behavior is expected to be present in clustering nonlinear subspaces as well, as we will see in the next experiments.

#### 5.1.2 Nonlinear subspace clustering

In this section, we analyze the nonlinear SC approaches on 4 well-known widely used nonlinear synthetic data sets in 2 dimensions: two half-kernels, two spirals, four corners and two arcs. For each data set, we sample 1000 total data points from the corresponding manifolds[13]. The data is normalized to have values between 0 and 1. The obtained connectivity graphs and the corresponding segmentation are shown in Figures 7, 8, 9 and 10. For each data set, the ground-truth clustering is shown and is followed by the obtained connectivity graphs of the nonlinear approaches. The average and standard deviation of clustering accuracy for 10 different sampling of the manifolds are reported in Table 5.

Table 5: The average and standard deviation of the accuracy, in percent, of SC approaches over 10 trials for nonlinear synthetic data sets. The best accuracy is indicated in bold, the second best is underlined

| | | SMCE | SMR | KSSC | LRL$\ell_1$-SSC | LKG | KNN-SSC | DSC-Net | DASC | SSC-L2 | LRR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Half-kernels | mean | **100** | 63.06 | **100** | 68.88 | 62.60 | <u>99.99</u> | 58.23 | 64.38 | 70.87 | 62.00 |
| | std | 0 | 2.05 | 0 | 1.10 | 2.96 | 0.03 | 4.08 | 8.58 | 2.08 | 2.20 |
| Two Spirals | mean | 98.68 | 64.64 | **100** | 51.05 | 68.02 | <u>98.99</u> | 55.58 | 60.66 | 63.98 | 64.84 |
| | std | 1.69 | 1.54 | 0 | 0.61 | 6.23 | 2.28 | 3.16 | 4.53 | 2.16 | 2.29 |
| Four Corners | mean | **100** | 65.19 | **100** | <u>99.98</u> | 99.43 | **100** | 48.08 | 58.48 | 58.23 | 62.97 |
| | std | 0 | 1.54 | 0 | 0.06 | 1.80 | 0 | 6.18 | 5.51 | 5.82 | 2.51 |
| Two Arcs | mean | 72.93 | 51.92 | 73.13 | **98.36** | 70.42 | <u>73.18</u> | 58.07 | 66.37 | 55.99 | 51.56 |
| | std | 1.57 | 1.51 | 1.35 | 0.91 | 6.39 | 2.06 | 6.23 | 7.72 | 1.71 | 1.27 |

We observe that:

- Linear SC algorithms, SSC-L2 and LRR do not perform well on nonlinear data sets. Similar result were obtained for SSC-L1, and hence we did not include them in Table 5.
- SMCE, KSSC with RBF kernel and KNN-SSC successfully produced sparse but well-connected graph on the half-kernels and 4 corners data sets.
- For the two spirals data set, the performances of SMCE and KNN-SSC are not consistent over the 10 different samplings of the manifolds. In particular, the inner ending points of the two spirals might not have subspace preserving representation and the performance might depend on the sampling process and the spectral clustering step.

[13]We used the code from https://www.mathworks.com/matlabcentral/fileexchange/41459-6-functions-for-generating-artificial-datasets.

- Both SMCE and KSSC fail to properly segment the two arcs data set. This is due to the intersection of the two nonlinear manifolds. Hence, we can conclude that *SMCE and KSSC might not be successful in segmenting intersecting manifolds.* Note that the rest of the connectivities are correct and sparse, the failure comes from the confusion at the points close to the intersection.

- SMR is not successful in clustering nonlinear manifolds. The coefficient matrix corresponding to SMR has many wrong connections. This suggests that, on top of the grouping property and the locality preserving regularizations, sparsity also plays a critical role.

- LR$\ell_1$-SSC is the only successful algorithm in clustering two arcs (although it produces many wrong connections). This algorithm is also successful in clustering the four corners data set. But as the nonlinearity increases (such as two spirals or half kernels), the clustering performance decreases significantly.

- The two arcs data set (see Figure 10) is an example of two intersecting manifolds; they intersect at (0.29,0.71). We have observed that if the two arcs intersect at (0,0) instead, then the clustering accuracy of KNN-SSC and SMR reaches 100%. This observation does not hold for other algorithms but this suggests that additional affinity constraints on the coefficient matrix might lead to better results in some cases.

- Neural network based approaches, that is, DSC-Net and DASC, perform very poorly. The reason is that their embedding acted as an almost identity mapping and merely rotated the input datta. In other words, they failed to produce a meaningful embedding in the encoder output. This is consistent with the results reported in [29].

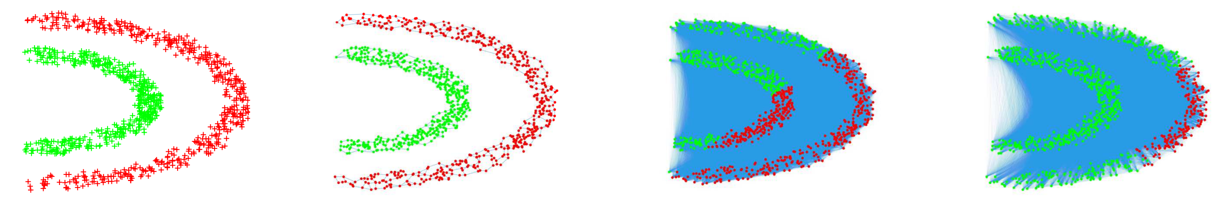
## 5.2 Real-world data sets

In this section, we compare the performances of SC algorithms on three widely used real-world data sets:

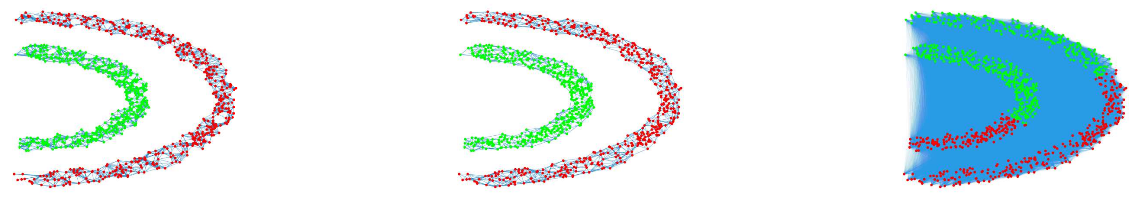
- **Extended Yale B [24, 55]:** The Extended Yale B data set contains 2,414 images of 38 individuals, taken under different illumination conditions. There are 64 frontal-face images, each of the size $192 \times 168$ pixels, per person. The images were downsampled to $48 \times 42$ pixels, similar to the general setting provided in [21]. Different number of clusters are considered to analyze the performance with respect to the number of subspaces.

- **Coil-20 [74]:** Columbia Object Image Library contains 72 images from 20 objects. The images were taken from objects in different poses. The images are of size $128 \times 128$ pixels and following [42], we downsampled them to $32 \times 32$ pixels.

- **MNIST test data set:** The MNIST data set contains 70,000 images of 10 handwritten digits where each image is of size 28-by-28 pixels. In particular, this data set includes training and test sets of 60,000 and 10,000 images, respectively. Due to high computational complexity of the majority of the nonlinear approaches (see Section 4), we only use the MNIST test data set for the evaluation, which is a common practice for evaluating computationally expensive clustering approaches [126, 127].

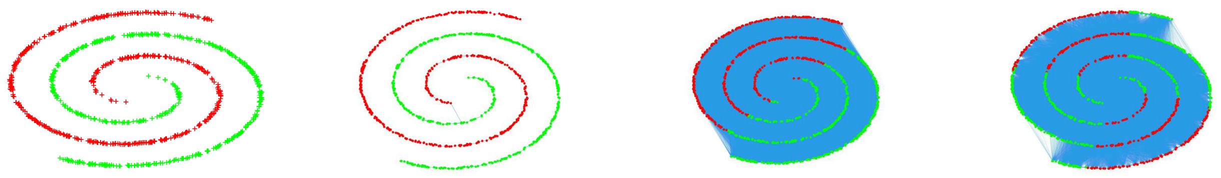


(a) ground-truth (b) SMCE (100%) (c) SMR (61.80%) (d) LR$\ell_1$-SSC (68.20%)

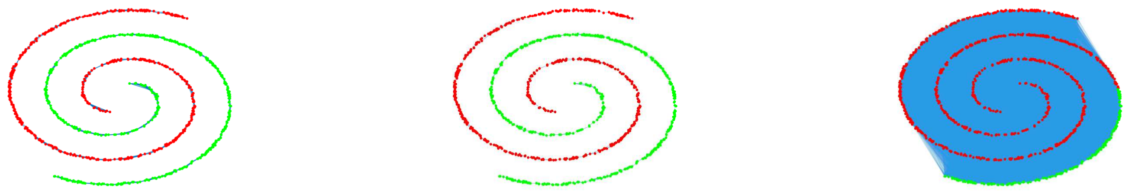


(e) KNN-SSC (100%) (f) KSSC (RBF) (100%) (g) LKG (61.50%)

Figure 7: Comparing the performance of nonlinear SC approaches on a half kernels synthetic data set. The accuracy of the displayed instances is indicated in brackets. The average accuracy over the 10 randomly generated instances can be found in Table 5.

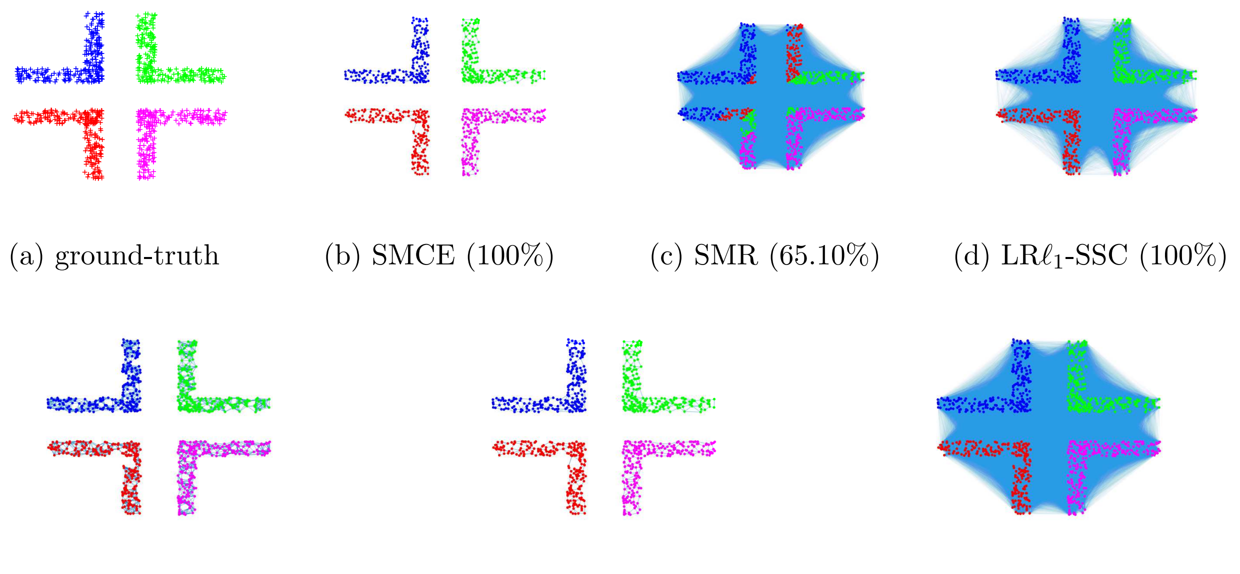


(a) ground-truth (b) SMCE (100.00%) (c) SMR (63.90%) (d) LR$\ell_1$-SSC (51.00%)



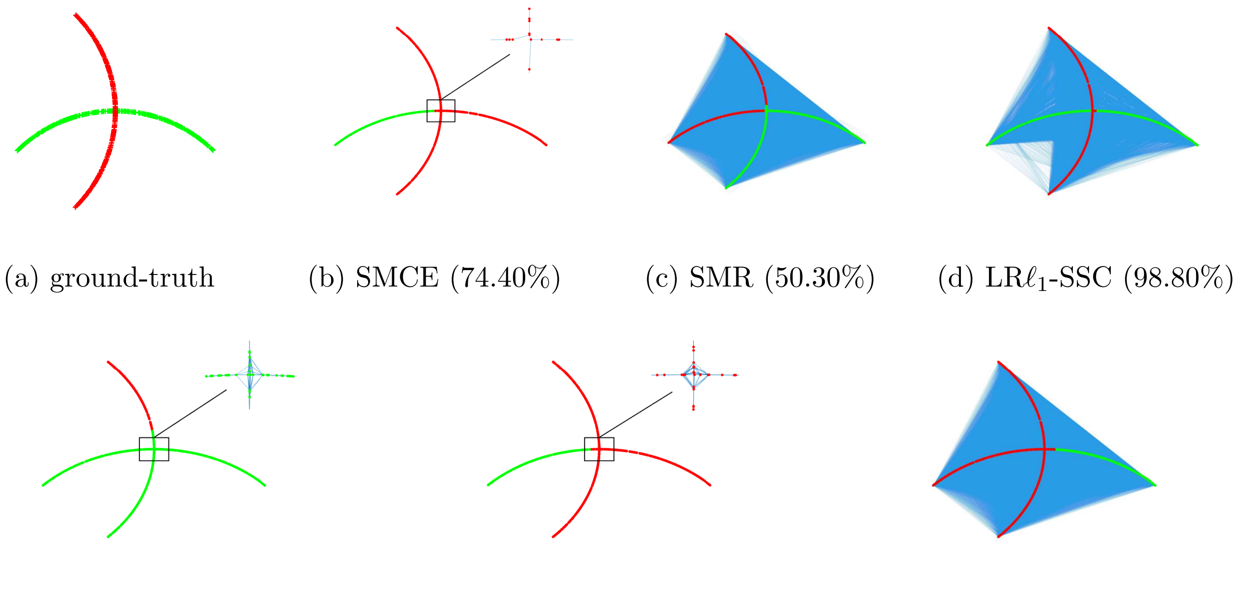
(e) KNN-SSC (100%) (f) KSSC (RBF) (100%) (g) LKG (68.50%)

Figure 8: Comparing the performance of nonlinear SC approaches on a two spirals synthetic data set. The accuracy of the displayed instances is indicated in brackets. The average accuracy over the 10 randomly generated instances can be found in Table 5.


(a) ground-truth (b) SMCE (100%) (c) SMR (65.10%) (d) LR$\ell_1$-SSC (100%)

(e) KNN-SSC (100%) (f) KSSC (RBF) (100%) (g) LKG (100%)

Figure 9: Comparing the performance of nonlinear SC approaches on a four corners synthetic data set. The accuracy of the displayed instances is indicated in brackets. The average accuracy over the 10 randomly generated instances can be found in Table 5.


(a) ground-truth (b) SMCE (74.40%) (c) SMR (50.30%) (d) LR$\ell_1$-SSC (98.80%)

(e) KNN-SSC (70.30%) (f) KSSC (RBF) (74.10%) (g) LKG (74.10%)

Figure 10: Comparing the performance of nonlinear SC approaches on a two arcs synthetic data set. The accuracy of the displayed instances is indicated in brackets. The average accuracy over the 10 randomly generated instances can be found in Table 5.

The parameters of each approach are set to the values introduced by the code of the original papers (if the code is available). For the approaches that are implemented by us, the parameters are tuned within the ranges proposed by the corresponding paper for the best result for each data set; see Appendix B for selected parameters of each approach. The results (namely, the accuracy and the NMI) on Extended Yale B for different number of subjects, COIL-20, and MNIST are summarized in Tables 6, 7, and 8, respectively. The value of M indicates that the corresponding algorithm ran out of 16 GB memory, and the numbers indicated with * correspond to the algorithms that did not converge within 6 hours.

We observe the following:

- On the Extended Yale B data set, SSC-L1 has the overall best performance. As the number of subjects increases, the neural network based approaches, including DSC-Net-L2 and DASC, have the overall second best performances. LKG has the worst performance, even worse than KSSC with a single kernel.

- The high performance of SSC-L1 on Extended Yale B highlights the importance of modeling noise and corruptions while clustering. Based on the experiments in [21] (see Section 7.2.1), after removing the sparse noise by applying RPCA [10] on images from each cluster separately, the clustering accuracy of SSC increases to 100%. This suggests that Extended Yale B can be well modeled by linear subspaces after the noise is eliminated.

- On the COIL-20 data set, SMCE performs the best followed by KSSC. The neural-network based approaches perform the worse. Note that, the high accuracy reported on their corresponding papers heavily relies on the post-processing step rather than the obtained coefficient matrix by the model [29].

- On the MNIST data set, KNN-SSC performs the best, followed closely by SSC-L2 and SMCE. KSSC performs better than neural-network based approaches. The Laplacian regularized LR$\ell_1$-SSC did not converge within 6 hours of timelimit. This is due to the fact that the iterative optimization process based on ADMM involves solving a Sylvester equation in each iteration which is very time consuming compared to solving a linear system in the original SSC algorithm. The same observation holds for LRR as well because of the computation of an SVD in each iteration of ADMM. The multiple-kernel based approach of LKG ran out of 16 GB memory which highlights the high computational cost of kernel approaches that rely on computing multiple pair-wise dense Gram matrices.

## 6 Discussion, challenges and future directions of research

Following the extensive numerical comparisons, in this section, we discuss nonlinear SC algorithms from different perspectives with an emphasis on the existing challenges and possible future research directions.

Table 6: Comparison of nonlinear SC algorithms on the Extended Yale B data set. The best value is highlighted in bold, the second best is underlined.

| Metric | SMCE | SMR | LR$\ell_1$-SSC | KNN-SSC | KSSC | LKG | DSC-L2 | DASC | SSC-L2 | SSC-L1 | LRR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 subjects | | | | | | | | | | | |
| ACC | **100** | 95.31 | 99.22 | **100** | 88.28 | 50.78 | 92.97 | 93.75 | 99.22 | **100** | 83.59 |
| NMI | **100** | 73.06 | 94.18 | **100** | 54.12 | 0.01 | 64.77 | 66.27 | 94.18 | **100** | 48.39 |
| 3 subjects | | | | | | | | | | | |
| ACC | 55.21 | 59.89 | 79.69 | 58.85 | 61.45 | 34.37 | 60.94 | 86.45 | 88.02 | **99.47** | 59.37 |
| NMI | 35.21 | 39.06 | 58.33 | 35.98 | 44.52 | 0.02 | 47.22 | 67.04 | 69.33 | **97.55** | 38.26 |
| 5 subjects | | | | | | | | | | | |
| ACC | 60.00 | 63.12 | 84.37 | 49.68 | 64.06 | 24.37 | 65.93 | 89.06 | 89.06 | **99.37** | 50.93 |
| NMI | 54.79 | 53.04 | 76.73 | 38.06 | 56.87 | 1.26 | 58.49 | 77.99 | 80.72 | **97.99** | 41.37 |
| 10 subjects | | | | | | | | | | | |
| ACC | 51.56 | 62.34 | 64.22 | 57.19 | 44.06 | 21.25 | 69.68 | 73.75 | 54.06 | **91.25** | 61.87 |
| NMI | 50.98 | 63.61 | 64.38 | 57.63 | 43.51 | 14.64 | 67.89 | 70.66 | 53.74 | **87.97** | 62.91 |
| 20 subjects | | | | | | | | | | | |
| ACC | 57.66 | 71.48 | 46.56 | 58.05 | 47.65 | 18.43 | 71.79 | 71.87 | 66.17 | **87.03** | 70.31 |
| NMI | 60.61 | 74.35 | 51.42 | 67.02 | 52.08 | 20.25 | 74.05 | 75.15 | 69.30 | **85.99** | 72.58 |
| 38 subjects | | | | | | | | | | | |
| ACC | 55.02 | 73.56 | 46.01 | 52.09 | 50.29 | 18.54 | 69.32 | **74.38** | 65.21 | 73.06 | 69.86 |
| NMI | 62.76 | 77.74 | 52.62 | 63.46 | 59.56 | 27.41 | 74.41 | **77.90** | 69.92 | 77.20 | 73.89 |

Table 7: Comparison of nonlinear SC algorithms on the COIL20 data set. The best value is highlighted in bold, the second best is underlined.

| Metric | SMCE | SMR | LR$\ell_1$-SSC | KNN-SSC | KSSC | LKG | DSC-L2 | DASC | SSC-L2 | SSC-L1 | LRR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ACC | **90.76** | 64.03 | 82.29 | 79.44 | 85.06 | 73.40 | 59.58 | 57.77 | 75.62 | 74.86 | 60.83 |
| NMI | **96.71** | 73.57 | 93.21 | 91.12 | 95.70 | 82.72 | 74.11 | 67.32 | 88.55 | 90.30 | 74.50 |

Table 8: Comparison of nonlinear SC algorithms on the MNIST data set. The best value is highlighted in bold, the second best is underlined. The symbol M indicates that the algorithm ran out of 16 GB memory.

| Metric | SMCE | SMR | LR$\ell_1$-SSC | KNN-SSC | KSSC | LKG | DSC-L2 | DASC | SSC-L2 | SSC-L1 | LRR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ACC | 68.77 | 54.64 | 55.08* | **70.48** | 64.32 | M | 55.46 | 46.28 | 70.12 | 59.03 | 43.92* |
| NMI | 73.06 | 54.97 | 62.26* | 73.98 | 70.99 | M | 56.46 | 44.51 | **74.89** | 71.06 | 35.50* |

Table 9: Advantages and disadvantages of representative nonlinear SC approaches

| Category | Advantages | Disadvantages |
|---|---|---|
| Locality preserving | • Convenient adaptation of classic linear SC<br>• Interpretable and intuitive<br>• supporting both independent and disjoint subspaces | • Dependent on the estimation of locality parameter<br>• Fixed locality estimation for all data points |
| Kernel Based | • Based on classical mathematical theory to implicitly transform data into higher dimensions<br>• Easily applicable to the majority of linear formulations via the kernel trick<br>• Supporting both independent and disjoint subspaces | • No guarantee that kernels lead to implicit feature space suitable for linear subspace clustering<br>• Difficulty in choosing the right kernel(s)<br>• Difficulty in understanding and interpreting results<br>• Memory inefficiency due to Gram matrix |
| Neural-network Based | • Learning the nonlinear transformation based on the data<br>• High capacity for learning complex data representations | • Ill-posed and might lead to trivial embeddings<br>• No theoretical guarantees<br>• Memory inefficiency due to self-expressiveness representation in latent space<br>• Difficult optimization and no general rule for selecting critical parameters such as the number of epochs used for training<br>• Failure to cluster disjoint subspaces |

## 6.1 Discussion on the performance of nonlinear SC algorithms

The main advantages and disadvantages of each category of nonlinear SC algorithms are summarized in Table 9. Based on the experimental results, no nonlinear approach is completely superior to the other nonlinear approaches.

Locality preserving approaches, which are based on the assumption that the manifolds are smooth and well sampled, are intuitive while they can be easily designed by adapting the linear models; see Section 3.1. However, many of the approaches in this category are sensitive to intersecting subspaces, and they depend on locality estimation parameters (such as $k$ in KNN based similarity matrix construction approaches).

Kernel-based nonlinear approaches can be designed as direct extensions of linear models as well (depending on the loss function, which is usually based on the Frobenius norm). However, there is no guarantee that the corresponding feature space is more suitable for linear models. Moreover, with no prior knowledge, choosing the right kernel and tuning its parameters is highly nontrivial. We noticed that even though learning weighted combination of multiple kernels might be expected to reduce the sensitivity to the kernel parameters, defining a *proper criterion* to adaptively choose the weights of each kernel is not straightforward for all data sets and applications.

With the growing interest in neural network and their recent success in learning complex data representations in many fields, deep SC methods have received considerable attention in the past few years. These approaches tend to *learn* the kernel embedding function instead of using the existing predefined kernels. However, Haeffele et al. [29] pointed out potential theoretical concerns for neural network based nonlinear SC approaches that rely on autoencoder regularization. It is theoretically argued that the underlying model is ill-posed and encouraging the latent representation to have a union of subspaces structure through the additional self-expressive loss term (or layer) is insufficient and leads to degenerate and trivial embeddings of the data in many cases. The potential degenerate embeddings is also noticed in deep k-means algorithm under the name of "scaling-down phenomenon" [22] where it is possible to make the clustering loss (for SC, the self-expressive loss) arbitrarily small without changing the reconstruction loss of the network. This problem can be avoided by several strategies such as regularizing the network weights or the norm of the embedded data. However, even using proper regularizations, the joint learning of embedding and self-expressive representation by autoencoder regularized nonlinear SC approaches can still lead to trivial data geometry in the embedding space. These issues are pointed out from a theoretical perspective, under the assumption that the auto-encoder is highly expressive. Intuitively, based on this assumption, the network can generate many possible embeddings of the data in the latent space, $Z_\Theta$, while the decoder is still capable of reconstructing the data accurately.

However, in practice and for both synthetic and real-world data sets, the performance of current neural-network based nonlinear SC approaches is often inferior to other nonlinear approaches. In fact, DSC-Net as the representative approach, strongly benefits from an ad-hoc post-processing step to improve the quality of the obtained coefficient matrix (see the second introduced post-processing step in Section 2.1.2). Our numerical experiments showed that DSC-Net does not necessarily lead to subspace preserving representations in many cases including the data drawn from independent subspaces. Hence, we believe that neural network based nonlinear SC through enforced self-expressive representation in latent space does not impose significant constraints on the geometric arrangement of the embedded points and appears to be not sufficient nor necessarily successful to recover the union of latent linear subspace structure.

## 6.2 Challenges and Future directions of research

Nonlinear subspace/manifold clustering is a challenging problem for which many approaches have been proposed. However, many challenges remain, including the following:

- Sensitivity to parameters: SC is an unsupervised problem with no available labelling information. The majority of algorithms discussed in this paper are found to be sensitive to their parameters. However, with no prior knowledge, setting these parameters is highly nontrivial and tricky. This is especially true for neural network based approaches that are very sensitive to the number of epochs for the pre-training and fine-tuning steps. In our experimental results, we fine-tuned the number of the epochs; however, we noticed that the performance can degrade significantly as the number of epochs increases. Multiple kernel learning approaches are also sensitive to their parameters; in particular to the lack of "good" criterion for choosing the weight of each Kernel. Developing algorithms with less sensitivity to their parameters (and ideally parameter free) is a general challenging research direction for unsupervised learning tasks.

- Scalabe algorithms: The majority of nonlinear SC based on self-expressiveness are computationally expensive; see Section 4. Locality preserving algorithms are usually based on estimating a local neighborhood graph which has at least the complexity of $O(n^2)$ and if the neighboorhood graph is not sparse, the memory requirement would be huge too. Computing the Gram matrix in kernel based approaches is another example of computational inefficiency. Except a few works that address scalability issue in deep self-expressive based SC approaches, e.g., [93, 142], nonlinear SC algorithms are not practical for data sets with more than 10,000 data points. Computing and storing the $n \times n$ coefficient matrix $C$ is specifically an important bottleneck in these approaches.

- Theoretical guarantees: In contrast to linear SC algorithms whose theoretical aspects are well studied and understood, nonlinear alternatives are mostly intuition driven. Except for the few works in locality preserving nonlinear approaches (such as [37]), there is no clear theoretical analysis on conditions for accurate subspace recovery for nonlinear subspaces. The lack of theoretical analysis is also evident in neural network based nonlinear SC approaches. Understanding the *black box* of neural networks is in general an ongoing research direction and combining these highly expressive models with SC based on mere intuition might result in trivial geometric embeddings [29] (see also the discussion in Section 6.1). Hence, understanding and analyzing nonlinear approaches from the theoretical standpoint can help avoiding ill-posed models that might be intuitively appealing at first.

- Non-image data extensions: Almost all nonlinear SC approaches are evaluated on image data sets such as images of faces, objects and handwritten digits. Investigating nonlinear structures in other data formats is another interesting research direction. There are selective approaches for subspace data on Grassmann manifolds such as [113, 111] or for symmetric positive definite (SPD) matrices such as [132, 33], but the major focus in the past years has been on image data.

- Robustness: The majority of nonlinear SC approaches rely on the Frobenius norm to measure the data fitting error, and hence are sensitive to gross corruptions such as occlusions, and the presence of outliers. Improving the robustness in such scenarios is definitely an important practical aspect.

- Clustering data with intersecting manifolds: The results on the synthetic data sets highlighted the difficulty for nonlinear SC approaches to cluster manifolds that are close to each other or intersect. The closer the manifolds are, the more the locality preserving approaches are sensitive to the locality controlling parameter, and similarly the kernel based approaches to the kernel parameters. Moreover, learning discriminating features to disentangle the multiple manifolds structure in the embedding space is harder for neural networks for data from closer manifolds. Hence, a general challenging research direction is to develop approaches to capture internal multiple manifold structures when they are spatially close in the ambient space.

## 7 Conclusion

In this paper, we presented a comprehensive overview of nonlinear subspace clustering (nonlinear SC) approaches, our main focus being on algorithms based on self-expressiveness. In Section 3, we provided a taxonomy for classifying nonlinear SC approaches into three broad categories: locality preserving, kernel based, and neural network based approaches. The approaches within each category were further divided into detailed subcategories and were thoroughly reviewed and summarized. In Section 4, we briefly discussed the computational cost of these approaches. In Section 5, the representative approaches within each (sub)-category were extensively compared on synthetic and real-world data sets. Based on the obtained results, Section 6 discussed the advantages and disadvantages of the different algorithms, and also elaborated on the current challenges for future research.

## References

[1] Abavisani, M., Patel, V.M.: Deep multimodal subspace clustering networks. IEEE Journal of Selected Topics in Signal Processing **12**(6), 1601–1614 (2018)

[2] Abdolali, M., Gillis, N., Rahmati, M.: Scalable and robust sparse subspace clustering using randomized clustering and multilayer graphs. Signal Processing **163**, 166–180 (2019)

[3] Bach, F., Jenatton, R., Mairal, J., Obozinski, G.: Optimization with sparsity-inducing penalties. arXiv preprint arXiv:1108.0775 (2011)

[4] Basri, R., Jacobs, D.W.: Lambertian reflectance and linear subspaces. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(2), 218–233 (2003)

[5] Bau, D., Zhu, J.Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., Torralba, A.: Seeing what a gan cannot generate. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4502–4511 (2019)

[6] Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. Advances in neural information processing systems **14**, 585–591 (2001)

[7] Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of machine learning research **7**(Nov), 2399–2434 (2006)

[8] Bellman, R.: Dynamic programming. Science **153**(3731), 34–37 (1966)

[9] Bradley, P.S., Mangasarian, O.L.: K-plane clustering. Journal of Global Optimization **16**(1), 23–32 (2000)

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

[11] Chen, Y., Li, C.G., You, C.: Stochastic sparse subspace clustering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4155–4164 (2020)

[12] Chen, Y., Yi, Z.: Locality-constrained least squares regression for subspace clustering. Knowledge-Based Systems **163**, 51–56 (2019)

[13] Chen, Y., Zhang, L., Yi, Z.: Subspace clustering using a low-rank constrained autoencoder. Information Sciences **424**, 27–38 (2018)

[14] Cheng, B., Yang, J., Yan, S., Fu, Y., Huang, T.S.: Learning with $\ell^1$-graph for image analysis. IEEE transactions on image processing **19**(4), 858–866 (2009)

[15] Costeira, J., Kanade, T.: A multi-body factorization method for motion analysis. In: Proceedings of IEEE International Conference on Computer Vision, pp. 1071–1076. IEEE (1995)

[16] Costeira, J.P., Kanade, T.: A multibody factorization method for independently moving objects. International Journal of Computer Vision **29**(3), 159–179 (1998)

[17] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol. 1, pp. 886–893. Ieee (2005)

[18] Daverman, R.J., Venema, G.: Embeddings in manifolds, vol. 106. American Mathematical Soc. (2009)

[19] Deng, T., Ye, D., Ma, R., Fujita, H., Xiong, L.: Low-rank local tangent space embedding for subspace clustering. Information Sciences **508**, 1–21 (2020)

[20] Elhamifar, E., Vidal, R.: Sparse manifold clustering and embedding. In: Advances in neural information processing systems, pp. 55–63 (2011)

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

[22] Fard, M.M., Thonet, T., Gaussier, E.: Deep k-means: Jointly clustering with k-means and learning representations. Pattern Recognition Letters **138**, 185–192 (2020)

[23] Gao, H., Nie, F., Li, X., Huang, H.: Multi-view subspace clustering. In: Proceedings of the IEEE international conference on computer vision, pp. 4238–4246 (2015)

[24] Georghiades, A., Belhumeur, P.: Illumination cone models for faces recognition under variable lighting and pose. IEEE Trans. Pattern Anal. Mach. Intelligence (23), 6 (1998)

[25] Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp. 315–323 (2011)

[26] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680 (2014)

[27] Graves, A., Bellemare, M.G., Menick, J., Munos, R., Kavukcuoglu, K.: Automated curriculum learning for neural networks. arXiv preprint arXiv:1704.03003 (2017)

[28] Gruber, A., Weiss, Y.: Multibody factorization with uncertainty and missing data using the em algorithm. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., vol. 1, pp. I–I. IEEE (2004)

[29] Haeffele, B.D., You, C., Vidal, R.: A critique of self-expressive deep subspace clustering. arXiv preprint arXiv:2010.03697 (2020)

[30] Hahnloser, R.H., Seung, H.S.: Permitted and forbidden sets in symmetric threshold-linear networks. In: Advances in neural information processing systems, pp. 217–223 (2001)

[31] Han, S., Huang, H., Qin, H., Yu, D.: Locality-preserving l1-graph and its application in clustering. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 813–818 (2015)

[32] Hastie, T., Simard, P.: Metrics and models for handwritten character recognition. In: Conference on Statistical Science Honouring the Bicentennial of Stefano Franscini's Birth, pp. 203–219. Springer (1997)

[33] Hechmi, S., Gallas, A., Zagrouba, E.: Multi-kernel sparse subspace clustering on the riemannian manifold of symmetric positive definite matrices. Pattern Recognition Letters **125**, 21–27 (2019)

[34] Heckel, R., Bölcskei, H.: Robust subspace clustering via thresholding. IEEE Transactions on Information Theory **61**(11), 6320–6342 (2015)

[35] Higham, N.J.: Accuracy and stability of numerical algorithms. SIAM (2002)

[36] Ho, J., Yang, M.H., Lim, J., Lee, K.C., Kriegman, D.: Clustering appearances of objects under varying illumination conditions. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., vol. 1, pp. I–I. IEEE (2003)

[37] Hu, H., Lin, Z., Feng, J., Zhou, J.: Smooth representation clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3834–3841 (2014)

[38] Huang, D., Wang, C.D., Wu, J.S., Lai, J.H., Kwoh, C.K.: Ultra-scalable spectral clustering and ensemble clustering. IEEE Transactions on Knowledge and Data Engineering **32**(6), 1212–1226 (2019)

[39] Huang, K., Aviyente, S.: Sparse representation for signal classification. In: Advances in neural information processing systems, pp. 609–616 (2007)

[40] Ji, P., Reid, I., Garg, R., Li, H., Salzmann, M.: Adaptive low-rank kernel subspace clustering. arXiv preprint arXiv:1707.04974 (2017)

[41] Ji, P., Salzmann, M., Li, H.: Efficient dense subspace clustering. In: IEEE Winter Conference on Applications of Computer Vision, pp. 461–468. IEEE (2014)

[42] Ji, P., Zhang, T., Li, H., Salzmann, M., Reid, I.: Deep subspace clustering networks. In: Advances in Neural Information Processing Systems, pp. 24–33 (2017)

[43] Jiang, L., Huang, D., Liu, M., Yang, W.: Beyond synthetic noise: Deep learning on controlled noisy labels. In: International Conference on Machine Learning, pp. 4804–4815. PMLR (2020)

[44] Jiang, L., Meng, D., Yu, S.I., Lan, Z., Shan, S., Hauptmann, A.: Self-paced learning with diversity. In: Advances in Neural Information Processing Systems, pp. 2078–2086 (2014)

[45] Jiang, Y., Yang, Z., Xu, Q., Cao, X., Huang, Q.: When to learn what: Deep cognitive subspace clustering. In: Proceedings of the 26th ACM international conference on Multimedia, pp. 718–726 (2018)

[46] Jolliffe, I.T.: Principal component analysis. Springer, New York (1986)

[47] Kanatani, K.i.: Motion segmentation by subspace separation and model selection. In: Proceedings Eighth IEEE International Conference on computer Vision. ICCV 2001, vol. 2, pp. 586–591. IEEE (2001)

[48] Kang, Z., Lu, X., Lu, Y., Peng, C., Chen, W., Xu, Z.: Structure learning with similarity preserving. Neural Networks (2020)

[49] Kang, Z., Lu, X., Yi, J., Xu, Z.: Self-weighted multiple kernel learning for graph-based clustering and semi-supervised classification. arXiv preprint arXiv:1806.07697 (2018)

[50] Kang, Z., Peng, C., Cheng, Q.: Twin learning for similarity and clustering: A unified kernel approach. arXiv preprint arXiv:1705.00678 (2017)

[51] Kang, Z., Wen, L., Chen, W., Xu, Z.: Low-rank kernel learning for graph-based clustering. Knowledge-Based Systems **163**, 510–517 (2019)

[52] Kang, Z., Zhao, X., Peng, C., Zhu, H., Zhou, J.T., Peng, X., Chen, W., Xu, Z.: Partition level multiview subspace clustering. Neural Networks **122**, 279–288 (2020)

[53] Kheirandishfard, M., Zohrizadeh, F., Kamangar, F.: Multi-level representation learning for deep subspace clustering. In: The IEEE Winter Conference on Applications of Computer Vision, pp. 2039–2048 (2020)

[54] Lane, C., Boger, R., You, C., Tsakiris, M., Haeffele, B., Vidal, R.: Classifying and comparing approaches to subspace clustering with missing data. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 0–0 (2019)

[55] Lee, K.C., Ho, J., Kriegman, D.J.: Acquiring linear subspaces for face recognition under variable lighting. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(5), 684–698 (2005)

[56] Li, C.G., You, C., Vidal, R.: On geometric analysis of affine sparse subspace clustering. IEEE Journal of Selected Topics in Signal Processing **12**(6), 1520–1533 (2018)

[57] Li, Z., Zhang, Y., Arora, S.: Why are convolutional nets more sample-efficient than fully-connected nets? arXiv preprint arXiv:2010.08515 (2020)

[58] Liang, J., Yang, J., Lee, H.Y., Wang, K., Yang, M.H.: Sub-gan: An unsupervised generative model via subspaces. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 698–714 (2018)

[59] Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., Ma, Y.: Robust recovery of subspace structures by low-rank representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(1), 171–184 (2012)

[60] Liu, J., Chen, Y., Zhang, J., Xu, Z.: Enhancing low-rank subspace clustering by manifold regularization. IEEE Transactions on Image Processing **23**(9), 4022–4030 (2014)

[61] Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the seventh IEEE international conference on computer vision, vol. 2, pp. 1150–1157. Ieee (1999)

[62] Lu, C., Feng, J., Lin, Z., Mei, T., Yan, S.: Subspace clustering by block diagonal representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(2), 487–501 (2018)

[63] Lu, C., Feng, J., Lin, Z., Mei, T., Yan, S.: Subspace clustering by block diagonal representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(2), 487–501 (2019). DOI 10.1109/TPAMI.2018.2794348

[64] Lu, C., Feng, J., Lin, Z., Yan, S.: Correlation adaptive subspace segmentation by trace lasso. In: Proceedings of the IEEE international conference on computer vision, pp. 1345–1352 (2013)

[65] Lu, C., Tang, J., Lin, M., Lin, L., Yan, S., Lin, Z.: Correntropy induced l2 graph for robust subspace clustering. In: Proceedings of the IEEE international conference on computer vision, pp. 1801–1808 (2013)

[66] Lu, C.Y., Min, H., Zhao, Z.Q., Zhu, L., Huang, D.S., Yan, S.: Robust and efficient subspace segmentation via least squares regression. In: European conference on computer vision, pp. 347–360. Springer (2012)

[67] Lu, X., Wang, Y., Yuan, Y.: Graph-regularized low-rank representation for destriping of hyperspectral images. IEEE transactions on geoscience and remote sensing **51**(7), 4009–4018 (2013)

[68] Lu, Z., Carreira-Perpinan, M.A.: Constrained spectral clustering through affinity propagation. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2008)

[69] Luo, D., Nie, F., Ding, C., Huang, H.: Multi-subspace representation and discovery. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 405–420. Springer (2011)

[70] Ma, X., Huang, H., Wang, Y., Romano, S., Erfani, S., Bailey, J.: Normalized loss functions for deep learning with noisy labels. In: International Conference on Machine Learning, pp. 6543–6553. PMLR (2020)

[71] Maggu, J., Majumdar, A., Chouzenoux, E., Chierchia, G.: Deeply transformed subspace clustering. Signal Processing p. 107628 (2020)

[72] Matsushima, S., Brbic, M.: Selective sampling-based scalable sparse subspace clustering. In: Advances in Neural Information Processing Systems, pp. 12,416–12,425 (2019)

[73] Nasihatkon, B., Hartley, R.: Graph connectivity in sparse subspace clustering. In: CVPR 2011, pp. 2137–2144. IEEE (2011)

[74] Nene, S.A., Nayar, S.K., Murase, H., et al.: Columbia object image library (coil-100) (1996)

[75] Ng, A.Y., Jordan, M.I., Weiss, Y., et al.: On spectral clustering: Analysis and an algorithm. Advances in neural information processing systems **2**, 849–856 (2002)

[76] Northcutt, C.G., Jiang, L., Chuang, I.L.: Confident learning: Estimating uncertainty in dataset labels. arXiv preprint arXiv:1911.00068 (2019)

[77] Ojala, T., Pietikainen, M., Harwood, D.: Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. In: Proceedings of 12th international conference on pattern recognition, vol. 1, pp. 582–585. IEEE (1994)

[78] Patel, V.M., Van Nguyen, H., Vidal, R.: Latent space sparse and low-rank subspace clustering. IEEE Journal of Selected Topics in Signal Processing **9**(4), 691–701 (2015)

[79] Patel, V.M., Vidal, R.: Kernel sparse subspace clustering. In: 2014 IEEE international conference on image processing (ICIP), pp. 2849–2853. IEEE (2014)

[80] Peng, X., Feng, J., Lu, J., Yau, W.Y., Yi, Z.: Cascade subspace clustering. In: Thirty-First AAAI conference on artificial intelligence (2017)

[81] Peng, X., Feng, J., Xiao, S., Yau, W.Y., Zhou, J.T., Yang, S.: Structured autoencoders for subspace clustering. IEEE Transactions on Image Processing **27**(10), 5076–5086 (2018)

[82] Peng, X., Feng, J., Zhou, J.T., Lei, Y., Yan, S.: Deep subspace clustering. IEEE Transactions on Neural Networks and Learning Systems (2020)

[83] Peng, X., Xiao, S., Feng, J., Yau, W.Y., Yi, Z.: Deep subspace clustering with sparsity prior. In: IJCAI, pp. 1925–1931 (2016)

[84] Peng, X., Yu, Z., Yi, Z., Tang, H.: Constructing the l2-graph for robust subspace learning and subspace clustering. IEEE transactions on cybernetics **47**(4), 1053–1066 (2016)

[85] Pourbahrami, S., Balafar, M.A., Khanli, L.M., Kakarash, Z.A.: A survey of neighborhood construction algorithms for clustering and classifying data points. Computer Science Review **38**, 100,315 (2020)

[86] Pourkamali-Anaraki, F., Folberth, J., Becker, S.: Efficient solvers for sparse subspace clustering. Signal Processing **172**, 107,548 (2020)

[87] Qiao, L., Zhang, L., Chen, S., Shen, D.: Data-driven graph construction and graph learning: A review. Neurocomputing **312**, 336–351 (2018)

[88] Ren, Z., Lei, H., Sun, Q., Yang, C.: Simultaneous learning coefficient matrix and affinity graph for multiple kernel clustering. Information Sciences (2020)

[89] Ren, Z., Li, H., Yang, C., Sun, Q.: Multiple kernel subspace clustering with local structural graph and low-rank consensus kernel learning. Knowledge-Based Systems **188**, 105,040 (2020)

[90] Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. science **290**(5500), 2323–2326 (2000)

[91] Saul, L.K., Roweis, S.T.: Think globally, fit locally: unsupervised learning of low dimensional manifolds. Journal of machine learning research **4**(Jun), 119–155 (2003)

[92] Sekmen, A., Koku, A.B., Parlaktuna, M., Abdul-Malek, A., Vanamala, N.: Unsupervised deep learning for subspace clustering. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 2089–2094. IEEE (2017)

[93] Seo, J., Koo, J., Jeon, T.: Deep closed-form subspace clustering. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 0–0 (2019)

[94] Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence **22**(8), 888–905 (2000)

[95] Soltanolkotabi, M., Candes, E.J., et al.: A geometric analysis of subspace clustering with outliers. The Annals of Statistics **40**(4), 2195–2238 (2012)

[96] Soltanolkotabi, M., Elhamifar, E., Candes, E.J., et al.: Robust subspace clustering. The Annals of Statistics **42**(2), 669–699 (2014)

[97] Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. Journal of machine learning research **3**(Dec), 583–617 (2002)

[98] Tierney, S., Guo, Y., Gao, J.: Efficient sparse subspace clustering by nearest neighbour filtering. arXiv preprint arXiv:1704.03958 (2017)

[99] Tipping, M.E., Bishop, C.M.: Mixtures of probabilistic principal component analyzers. Neural computation **11**(2), 443–482 (1999)

[100] Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. International journal of computer vision **9**(2), 137–154 (1992)

[101] Traganitis, P.A., Giannakis, G.B.: Sketched subspace clustering. IEEE Transactions on Signal Processing **66**(7), 1663–1675 (2017)

[102] Tremblay, N., Puy, G., Gribonval, R., Vandergheynst, P.: Compressive spectral clustering. In: International conference on machine learning, pp. 1002–1011. PMLR (2016)

[103] Tsakiris, M.C., Vidal, R.: Algebraic clustering of affine subspaces. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(2), 482–489 (2017)

[104] Tseng, P.: Nearest q-flat to m points. Journal of Optimization Theory and Applications **105**(1), 249–252 (2000)

[105] Udell, M., Townsend, A.: Why are big data matrices approximately low rank? SIAM Journal on Mathematics of Data Science **1**(1), 144–160 (2019)

[106] Van Der Maaten, L., Postma, E., Van den Herik, J.: Dimensionality reduction: A comparative review. Tech. Rep. TiCC-TR 2009-005, Tilburg University (2009)

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

[108] Vidal, R., Ma, Y., Sastry, S.: Generalized principal component analysis (gpca). IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(12), 1945–1959 (2005)

[109] Vidal, R., Ma, Y., Sastry, S.S.: Principal component analysis. In: Generalized principal component analysis. Springer (2016)

[110] Von Luxburg, U.: A tutorial on spectral clustering. Statistics and computing **17**(4), 395–416 (2007)

[111] Wang, B., Hu, Y., Gao, J., Sun, Y., Yin, B.: Kernelized LRR on Grassmann manifolds for subspace clustering. arXiv preprint arXiv:1601.02124 (2016)

[112] Wang, B., Hu, Y., Gao, J., Sun, Y., Yin, B.: Laplacian LRR on product Grassmann manifolds for human activity clustering in multicamera video surveillance. IEEE Transactions on Circuits and Systems for Video Technology **27**(3), 554–566 (2016)

[113] Wang, B., Hu, Y., Gao, J., Sun, Y., Yin, B.: Localized LRR on grassmann manifold: An extrinsic view. IEEE Transactions on Circuits and Systems for Video Technology **28**(10), 2524–2536 (2017)

[114] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: 2010 IEEE computer society conference on computer vision and pattern recognition, pp. 3360–3367. IEEE (2010)

[115] Wang, S., Yuan, X., Yao, T., Yan, S., Shen, J.: Efficient subspace segmentation via quadratic programming. In: Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)

[116] Wang, Y.X., Xu, H.: Noisy sparse subspace clustering. The Journal of Machine Learning Research **17**(1), 320–360 (2016)

[117] Wang, Y.X., Xu, H., Leng, C.: Provable subspace clustering: When LRR meets SSC. In: Advances in Neural Information Processing Systems, pp. 64–72 (2013)

[118] Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(2), 210–227 (2008)

[119] Xiang, S., Nie, F., Zhang, C.: Learning a mahalanobis distance metric for data clustering and classification. Pattern recognition **41**(12), 3600–3612 (2008)

[120] Xiao, S., Tan, M., Xu, D., Dong, Z.Y.: Robust kernel low-rank representation. IEEE transactions on neural networks and learning systems **27**(11), 2268–2281 (2015)

[121] Xie, Y., Liu, J., Qu, Y., Tao, D., Zhang, W., Dai, L., Ma, L.: Robust kernelized multiview self-representation for subspace clustering. IEEE Transactions on Neural Networks and Learning Systems (2020)

[122] Xue, X., Zhang, X., Feng, X., Sun, H., Chen, W., Liu, Z.: Robust subspace clustering based on non-convex low-rank approximation and adaptive kernel. Information Sciences **513**, 190–205 (2020)

[123] Yan, D., Huang, L., Jordan, M.I.: Fast approximate spectral clustering. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 907–916 (2009)

[124] Yang, C., Ren, Z., Sun, Q., Wu, M., Yin, M., Sun, Y.: Joint correntropy metric weighting and block diagonal regularizer for robust multiple kernel subspace clustering. Information Sciences **500**, 48–66 (2019)

[125] Yang, J., Liang, J., Wang, K., Rosin, P.L., Yang, M.H.: Subspace clustering via good neighbors. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(6), 1537–1544 (2019)

[126] Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5147–5156 (2016)

[127] Yang, X., Deng, C., Zheng, F., Yan, J., Liu, W.: Deep spectral clustering using dual autoencoder network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4066–4075 (2019)

[128] Yang, Y., Feng, J., Jojic, N., Yang, J., Huang, T.S.: $\ell^0$-sparse subspace clustering. In: European conference on computer vision, pp. 731–747. Springer (2016)

[129] Yang, Y., Wang, Z., Yang, J., Wang, J., Chang, S., Huang, T.S.: Data clustering by laplacian regularized l1-graph. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, pp. 3148–3149 (2014)

[130] Yankelevsky, Y., Elad, M.: ADMM pursuit for manifold regularized sparse coding

[131] Yin, M., Gao, J., Lin, Z., Shi, Q., Guo, Y.: Dual graph regularized latent low-rank representation for subspace clustering. IEEE Transactions on Image Processing **24**(12), 4918–4933 (2015)

[132] Yin, M., Guo, Y., Gao, J., He, Z., Xie, S.: Kernel sparse subspace clustering on symmetric positive definite manifolds. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5157–5164 (2016)

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

[134] You, C., Li, C.G., Robinson, D.P., Vidal, R.: Oracle based active set algorithm for scalable elastic net subspace clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3928–3937 (2016)

[135] You, C., Robinson, D., Vidal, R.: Scalable sparse subspace clustering by orthogonal matching pursuit. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3918–3927 (2016)

[136] You, C., Vidal, R.: Geometric conditions for subspace-sparse recovery. In: International Conference on Machine Learning, pp. 1585–1593 (2015)

[137] Yu, Z., Zhang, Z., Cao, W., Liu, C., Chen, J.P., San Wong, H.: Gan-based enhanced deep subspace clustering networks. IEEE Transactions on Knowledge and Data Engineering (2020)

[138] Zhang, G.Y., Zhou, Y.R., He, X.Y., Wang, C.D., Huang, D.: One-step kernel multi-view subspace clustering. Knowledge-Based Systems **189**, 105,126 (2020)

[139] Zhang, G.Y., Zhou, Y.R., Wang, C.D., Huang, D., He, X.Y.: Joint representation learning for multi-view subspace clustering. Expert Systems with Applications p. 113913 (2020)

[140] Zhang, J., Li, C.G., You, C., Qi, X., Zhang, H., Guo, J., Lin, Z.: Self-supervised convolutional subspace clustering network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5473–5482 (2019)

[141] Zhang, L., Yang, M., Feng, X.: Sparse representation or collaborative representation: Which helps face recognition? In: 2011 International conference on computer vision, pp. 471–478. IEEE (2011)

[142] Zhang, T., Ji, P., Harandi, M., Hartley, R., Reid, I.: Scalable deep k-subspace clustering. In: Asian Conference on Computer Vision, pp. 466–481. Springer (2018)

[143] Zhang, T., Ji, P., Harandi, M., Huang, W., Li, H.: Neural collaborative subspace clustering. arXiv preprint arXiv:1904.10596 (2019)

[144] Zhang, T., Ji, R., Liu, W., Tao, D., Hua, G.: Semi-supervised learning with manifold fitted graphs. In: Twenty-Third International Joint Conference on Artificial Intelligence. Citeseer (2013)

[145] Zhang, X., Sun, H., Liu, Z., Ren, Z., Cui, Q., Li, Y.: Robust low-rank kernel multi-view subspace clustering based on the schatten p-norm and correntropy. Information Sciences **477**, 430–447 (2019)

[146] Zhang, Z., Xu, Y., Yang, J., Li, X., Zhang, D.: A survey of sparse representation: algorithms and applications. IEEE access **3**, 490–530 (2015)

[147] Zheng, Y., Zhang, X., Xu, Y., Qin, M., Ren, Z., Xue, X.: Robust multi-view subspace clustering via weighted multi-kernel learning and co-regularization. IEEE Access **8**, 113,030–113,041 (2020)

[148] Zheng, Y., Zhang, X., Yang, S., Jiao, L.: Low-rank representation with local constraint for graph construction. Neurocomputing **122**, 398–405 (2013)

[149] Zhong, G., Pun, C.M.: Nonnegative self-representation with a fixed rank constraint for subspace clustering. Information Sciences **518**, 127–141 (2020)

[150] Zhou, L., Wang, S., Bai, X., Zhou, J., Hancock, E.: Iterative deep subspace clustering. In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), pp. 42–51. Springer (2018)

[151] Zhou, L., Xiao, B., Liu, X., Zhou, J., Hancock, E.R., et al.: Latent distribution preserving deep subspace clustering. In: 28th International Joint Conference on Artificial Intelligence. York (2019)

[152] Zhou, P., Hou, Y., Feng, J.: Deep adversarial subspace clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1596–1604 (2018)

[153] Zhu, W., Lu, J., Zhou, J.: Nonlinear subspace clustering for image clustering. Pattern Recognition Letters **107**, 131–136 (2018)

[154] Zhu, W., Peng, B.: Sparse and low-rank regularized deep subspace clustering. Knowledge-Based Systems **204**, 106,199 (2020)

[155] Zhuang, L., Wang, J., Lin, Z., Yang, A.Y., Ma, Y., Yu, N.: Locality-preserving low-rank representation for graph construction from nonlinear manifolds. Neurocomputing **175**, 715–722 (2016)

## A Synthetic data generation

In this section, we provide the detailed data generation process for independent and disjoint linear subspaces used in Section 5.1.1.

We introduce a semi-random model for generating the linear synthetic data. In this model, the bases of the subspaces are fixed, to control the angle between them, but the data points are generated at random from each of the subspaces. The data generation model for independent and disjoint subspaces are as follows:

- **Independent subspaces:** In order to generate two independent subspaces with intrinsic dimension $m$ in a $d$-dimensional space ($m < d$) with controlled affinity between subspaces, the basis of the subspaces are initially constructed in $2m$-dimensional space as follows:

$$\hat{U_1} = \begin{pmatrix} I_m \\ 0_m \end{pmatrix}, \hat{U_2} = \begin{pmatrix} \cos(\theta) I_m \\ \sin(\theta) I_m \end{pmatrix},$$

  where $I_m$ and $0_m$ are the identity and zero $m \times m$ matrices and $\theta \in [0, \frac{\pi}{2}]$. The affinity between the two subspaces is explicitly controlled by the parameter of $\theta$. By decreasing the value of $\theta$ from $\frac{\pi}{2}$ to 0, the affinity between the subspaces decreases and the subspace clustering task gets more challenging [95].

  Let $N$ be the total number of data points on all subspaces, with half of the data points on each subspace. The data points within each subspace in $2m$-dimensional space are randomly produced by setting linear mixture weights at random using the Gaussian distribution and multiplying them by the initial bases (in Matlab $\hat{X}_i = \hat{U}_i * randn(d, N/2)$ for $i = 1, 2$). Let $\hat{X}_1$ and $\hat{X}_2$ denote the $2m$-dimensional data points generated for each subspace. The $2m$-dimensional data points are transferred to the final $d$-dimensional space by multiplying them by orthogonal columns of random matrix $P \in \mathbb{R}^{d \times 2m}$ (in Matlab, $P$ is generated by $orth(randn(d, 2m))$):

$$X_i = P \times \hat{X}_i, \text{ for } i = 1, 2.$$

  Note that, by increasing the dimension using the orthogonal matrix $P$, we keep the affinity between subspaces in the initial $2m$-dimensional space which is controlled by the parameter $\theta$. It is a common practice to normalize the data points to have unit $\ell_2$ norm, however, since normalization ruins the structure for *affine* and *nonlinear* subspaces, we do not normalize the data.

- **Disjoint subspaces:** For disjoint synthetic subspaces, we generate $N$ samples from three subspaces (with $N/3$ samples for each subspace) similar to the independent case. In particular, three initial subspaces bases, with intrinsic dimension of $m$, are constructed in $2m$-dimensional space as:

$$\hat{U_1} = \begin{pmatrix} I_m \\ 0_m \end{pmatrix}, \hat{U_2} = \begin{pmatrix} \cos(\theta) I_m \\ \sin(\theta) I_m \end{pmatrix}, \hat{U_3} = \begin{pmatrix} \cos(\theta) I_m \\ -\sin(\theta) I_m \end{pmatrix}.$$

  Identical to the previous independent subspaces generation, the initial $2m$-dimensional data points within each subspace, that is, $\{\hat{X}_i\}_{i=1}^3$ are obtained by multiplying $\{\hat{U}_i\}_{i=1}^3$ by linear combinations produced from Gaussian distribution with zero mean and standard deviation of 1. The dimension of the data points are lifted to the final $d$-dimensional space by multiplying an orthogonal randomly generated matrix $P$.

## B Parameter setting

In this section, the selected parameters for each approach is listed in details in Table 10 for SMCE, SMR, LR$\ell_1$-SSC, KNN-SSC, KSSC, LKG, SSC-L2, SSC-L1 and LRR; in Table 11 for DSC-Net and in Table 12 for DASC.

The adaptive parameter selection in [21] is followed for setting the regularization parameter $\lambda$ which controls the importance of the self-expressiveness term, $||X - XC||_F^2$ (or $||\Phi(X) - \Phi(X)C||_F^2$ in kernel based approaches). To this end, the following data-driven strategy is used: $\lambda = \alpha \max_i \frac{1}{\max_{j \neq i} |X(:,j)^\top X(:,i)|}$. Hence, the larger the parameter $\alpha$ is, the more important the self-expressiveness term is considered in the optimization. This parameter setting is used for LR$\ell_1$-SSC, KNN-SSC, KSSC, SSC-L1 and SSC-L2.

Table 10: Parameters of the compared approaches.

| Approach | Parameters | | | | |
|---|---|---|---|---|---|
| | Linear | Nonlinear | Yale B | Coil 20 | MNIST |
| SMCE | $\lambda = 10$ | $\lambda = 20$ | $\lambda = 10$ | $\lambda = 20$ | $\lambda = 5$ |
| SMR | $\lambda = 1,\ k = 10$ | $\lambda = 0.1,\ k = 10$ | $\lambda = 1e3,\ k = 100$ | $\lambda = 1e3,\ k = 50$ | $\lambda = 1e5,\ k = 20$ |
| LR$\ell_1$-SSC | $\alpha = 20, \lambda_2 = 1,\ k = 10$ | $\alpha = 20, \lambda_2 = 10,\ k = 10$ | $\alpha = 10, \lambda_2 = 200,\ k = 100$ | $\alpha = 10, \lambda_2 = 20,\ k = 20$ | $\alpha = 10, \lambda_2 = 20,\ k = 20$ |
| KNN-SSC | $\alpha = 10, k = 10$ | $\alpha = 10, k = 10$ | $\lambda = 0.2, k = 100$ | $\lambda = 0.2, k = 20$ | $\lambda = 0.033, k = 100$ |
| KSSC | $\sigma = 1, \alpha = 20$ | $\sigma = 0.05, \alpha = 5$ | $\sigma = 10, \alpha = 10$ | $\sigma = 5, \alpha = 10$ | $\sigma = 5, \alpha = 10$ |
| LKG | $\lambda_1 = 0.1, \lambda_2 = 10,\ \lambda_3 = 1$ | $\lambda_1 = 0.1, \lambda_2 = 10,\ \lambda_3 = 1$ | $\lambda_1 = 0.1, \lambda_2 = 10,\ \lambda_3 = 0.5$ | $\lambda_1 = 0.1, \lambda_2 = 10,\ \lambda_3 = 0.5$ | $\lambda_1 = 0.1, \lambda_2 = 10, \lambda_3 = 0.5$ |
| SSC-L2 | $\alpha = 10$ | $\alpha = 20$ | $\alpha = 5$ | $\alpha = 5$ | $\alpha = 5$ |
| SSC-L1 | $\alpha = 5$ | $\alpha = 10$ | $\alpha = 20$ | $\alpha = 20$ | $\alpha = 20$ |
| LRR | $\lambda = 0.1$ | $\lambda = 0.1$ | $\lambda = 0.009$ | $\lambda = 0.009$ | $\lambda = 0.0092$ |

Table 11: Parameters of DSC-Net.

| data set | Parameters |
|---|---|
| Linear | 3-layered fully connected autoencoder: {10,8,4} units for encoder, $\lambda_1 = 1$, $\lambda_2 = 10^{c/10-3}$, # of epochs: $50 + 25c$ |
| Nonlinear | 3-layered fully connected autoencoder: {8,4,2} units for encoder, $\lambda_1 = 1$, $\lambda_2 = 10^{c/10-3}$, # of epochs: $50 + 25c$ |
| Yale B | 3-layered convolutional autoencoder: $\begin{matrix} encoder-1 \ \& \ decoder-3: 5\times 5 kernel, 10\ channels \\ encoder-2 \ \& \ decoder-2: 3\times 3 kernel, 20\ channels \\ encoder-3 \ \& \ decoder-1: 3\times 3 kernel, 30\ channels \end{matrix}$, $\lambda_1 = 1$, $\lambda_2 = 10^{c/10-3}$, # of epochs: $50 + 25c$ |
| Coil 20 | 1-layered convolutional autoencoder: $encoder-1 \ \& \ decoder-1: 3\times 3 kernel, 15\ channels$, $\lambda_1 = 1$, $\lambda_2 = 150$, # of epochs: 30 |
| MNIST | 3-layered convolutional autoencoder: $\begin{matrix} encoder-1 \ \& \ decoder-3: 5\times 5 kernel, 10\ channels \\ encoder-2 \ \& \ decoder-2: 3\times 3 kernel, 20\ channels \\ encoder-3 \ \& \ decoder-1: 3\times 3 kernel, 30\ channels \end{matrix}$, $\lambda_1 = 1$, $\lambda_2 = 1$, # of epochs: 1000 |

Table 12: Parameters of DASC.

| data set | Parameters |
|---|---|
| Linear | 2-layered fully connected autoencoder: {8,4} units for encoder, $\lambda_1 = 0.5$, $\lambda_2 = 0.1$, $\lambda_3 = 1$, # of pretrain epochs = 200, # of epochs: 300 |
| Nonlinear | 2-layered fully connected autoencoder: {2,2} units for encoder, $\lambda_1 = 0.5$, $\lambda_2 = 0.1$, $\lambda_3 = 1$, # of pretrain epochs = 200, # of epochs: 300 |
| Yale B | 3-layered convolutional autoencoder: $\begin{matrix} encoder-1 \ \& \ decoder-3: 5\times 5 kernel, 10\ channels \\ encoder-2 \ \& \ decoder-2: 3\times 3 kernel, 20\ channels \\ encoder-3 \ \& \ decoder-1: 3\times 3 kernel, 30\ channels \end{matrix}$, $\lambda_1 = 0.5$, $\lambda_2 = 0.1$, $\lambda_3 = 1$, # of pretrain epochs = 1000, # of epochs: 1500 |
| Coil 20 | 1-layered convolutional autoencoder: $encoder-1 \ \& \ decoder-1: 3\times 3 kernel, 15\ channels$, $\lambda_1 = 0.5$, $\lambda_2 = 0.1$, $\lambda_3 = 1$, # of pretrain epochs = 1000, # of epochs: 1200 |
| MNIST | 3-layered convolutional autoencoder: $\begin{matrix} encoder-1 \ \& \ decoder-3: 5\times 5 kernel, 10\ channels \\ encoder-2 \ \& \ decoder-2: 3\times 3 kernel, 20\ channels \\ encoder-3 \ \& \ decoder-1: 3\times 3 kernel, 30\ channels \end{matrix}$, $\lambda_1 = 0.5$, $\lambda_2 = 0.1$, $\lambda_3 = 1$, # of pretrain epochs = 1000, # of epochs: 2000 |