Kernel truncated regression representation for robust subspace clustering

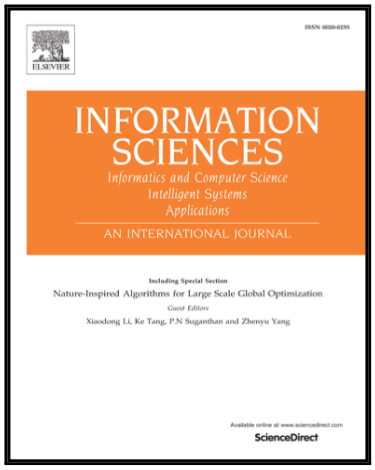
Zhen, Liangli; Peng, Dezhong; Wang, Wei; Yao, Xin

DOI:
10.1016/j.ins.2020.03.033

Document Version
Peer reviewed version

Citation for published version (Harvard):
Zhen, L, Peng, D, Wang, W & Yao, X 2020, 'Kernel truncated regression representation for robust subspace clustering', *Information Sciences*, vol. 524, pp. 59-76. https://doi.org/10.1016/j.ins.2020.03.033

Journal Pre-proof



Kernel Truncated Regression Representation for Robust Subspace Clustering

Liangli Zhen, Dezhong Peng, Wei Wang, Xin Yao

PII: S0020-0255(20)30212-7
DOI: https://doi.org/10.1016/j.ins.2020.03.033
Reference: INS 15293

To appear in: *Information Sciences*

Received date: 23 August 2019
Revised date: 26 February 2020
Accepted date: 11 March 2020

Please cite this article as: Liangli Zhen, Dezhong Peng, Wei Wang, Xin Yao, Kernel Truncated Regression Representation for Robust Subspace Clustering, *Information Sciences* (2020), doi: https://doi.org/10.1016/j.ins.2020.03.033

Highlights

- We propose a novel robust subspace clustering method, termed kernel truncated regression representation (KTRR). It can cluster the data points drawn from multiple nonlinear subspaces.
- A closed-form solution to KTRR is provided. The solution or the proposed model is a function of the kernel matrix and not directly related to the input data. It makes our method very efficient, especially when handling the problem that has high-dimensional input data.
- The proposed KTRR is further extended to handle large-scale data sets. We transform the scalability issue in KTRR as a kind of out-of-sample problem, and solve it with a "sampling, clustering, and classifying" strategy.
- We apply the KTRR method for several image clustering problems. Extensive experiments are conducted to investigate the effectiveness and efficiency of KTRR. The empirical results show that our method significantly outperforms current state-of-the-art subspace clustering algorithms in terms of accuracy, robustness, and computational cost

Kernel Truncated Regression Representation for Robust Subspace Clustering

Liangli Zhen[a], Dezhong Peng[b,c,*], Wei Wang[b], Xin Yao[d,e]

[a]*Institute of High Performance Computing, A*STAR, Singapore 138632*
[b]*College of Computer Science, Sichuan University, Chengdu 610065, China*
[c]*Peng Cheng Laboratory, Shenzhen 518055, China*
[d]*Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China*
[e]*CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK.*

Abstract

Subspace clustering aims to group data points into multiple clusters of which each corresponds to one subspace. Most existing subspace clustering approaches assume that input data lie on linear subspaces. In practice, however, this assumption usually does not hold. To achieve nonlinear subspace clustering, we propose a novel method, called kernel truncated regression representation. Our method consists of the following four steps: 1) projecting the input data into a hidden space, where each data point can be linearly represented by other data points; 2) calculating the linear representation coefficients of the data representations in the hidden space; 3) truncating the trivial coefficients to achieve robustness and block-diagonality; and 4) executing the graph cutting operation on the coefficient matrix by solving a graph Laplacian problem. Our method has the advantages of a closed-form solution and the capacity of clustering data points that lie on nonlinear subspaces. The first advantage makes our method efficient in handling large-scale datasets, and the second one enables the proposed method to conquer the nonlinear subspace clustering challenge. Extensive experiments on six benchmarks demonstrate the effectiveness and the efficiency of the proposed method in comparison with current state-of-the-art approaches.

*Corresponding author
Email address: pengdz@scu.edu.cn (Dezhong Peng)

Keywords: Kernel truncated regression; nonlinear subspace clustering; spectral clustering; kernel techniques.

1. Introduction

Subspace clustering is one of the most popular techniques for data analysis, which has attracted increasing interests of researchers from various areas, such as computer vision, image analysis, and signal processing [27]. With the assumption that high-dimensional data are approximately drawn from a union of low-dimensional subspaces, subspace clustering aims to seek a set of subspaces to fit a given data set and performs clustering based on the identified subspaces [39].

During the past decades, numerous approaches have been developed for subspace clustering, which can be roughly classified into the following four categories: iterative approaches [8], statistical approaches [34], algebraic approaches [40] and spectral clustering-based approaches [41, 48]. In recent years, spectral clustering-based approaches have attracted more attention and achieved state-of-the-art performance in image clustering and motion segmentation [49]. The key of this kind of approaches is to find a block-diagonal affinity matrix, where the element of the matrix denotes the similarity between two corresponding data points and the block-diagonal structure means that only the similarity between two intra-cluster data points is nonzero.

To obtain a block-diagonal affinity matrix, some researchers proposed to measure the similarity using the self-expression strategy in spectral clustering methods. Specifically, they represent each data point as a linear combination of the other points in the dataset itself and then use the representation coefficients to build the affinity matrix. The key difference among those methods is the constraint conducted on the representation coefficients and/or the manner of handling noises. For example, sparse subspace clustering (SSC) [7] assumes that each data point can be linearly represented by fewest other points and places the ℓ_1-norm minimisation constraint on the coefficient vectors. Low-rank

representation (LRR) [22] encourages the coefficient matrix to be low rank for capturing the global structure of the input database. To obtain the low rankness, LRR enforces a nuclear-norm minimisation constraint on the coefficient matrix. Different from SSC and LRR, least squares regression (LSR) [24] and truncated regression representation (TRR) [32] take ℓ_2-norm instead of the ℓ_1-norm and the nuclear-norm to constrain the representation coefficients. The main difference between LSR and TRR is that TRR has a truncated operation on the representation coefficient matrix. Most of these approaches handle the noise by minimising the ℓ_1-/ℓ_2-norm of the reconstruction error. To handle the complex noise, He *et al.* [11] proposed to maximise the correntropy between a given data point and its reconstruction with other points, and Wang *et al.* [42] proposed to minimise the entropy of the error between observation signal and its estimation. In addition, Li *et al.* [20] proposed to exploit the intrinsic geometric structure of data and the local and global structural consistencies over labels to learn discriminative features. Kang *et al.* [15] proposed to remove the noise and errors from the raw data adaptively using low-rank recovery and robust principal component analysis to achieve robust graph learning. Furthermore, from the point view of semi-supervised learning, [21] explores both the labelled and unlabelled to explicitly learn the block-diagonal structure in a nonnegative matrix factorization (NMF) framework. In recent years, some research works are developed to achieve clustering on the large-scale datasets. For instance, Peng *et al.* [33] extended SSC by using the sampling strategy. Kang *et al.* [17] used a smaller graph to approximate the full graph adaptively by learning from the raw data. These methods have shown promising performance in subspace clustering.

Note that these subspace clustering methods are originally developed to handle the data that are (approximately) drawn from a union of linear subspaces. They may not be able to obtain a satisfactory clustering result when the input data points lie on a set of nonlinear subspaces. In real-world systems, however, most of the collected data are located on nonlinear subspaces [37, 16]. It brings a challenge to linear subspace clustering methods and limits their applications

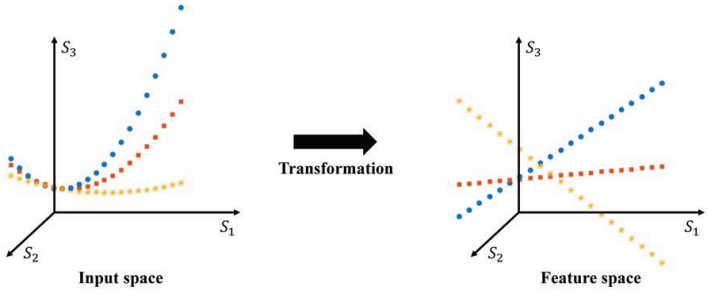


Figure 1: The basic idea of our method. By projecting the data into another space with an implicit nonlinear transformation, our method could solve the problem of nonlinear subspace clustering. The left and right plots correspond to the data distributions in the input and hidden space, respectively.

in the real world. Even though there are some deep learning-based clustering approaches [29] are developed to tackle this challenge, they usually need a large amount of data to be available and have high computational complexity, which hinders their applications largely.

To cluster the data drawn from a union of nonlinear subspaces, in this paper, we propose a novel robust subspace clustering method, termed kernel truncated regression representation (KTRR). Our basic idea is based on the assumption that there exists a hidden space in which the data can be linearly represented by each other. To illustrate this simple but effective idea, we provide a toy example in Fig. 1. From the example, we can see that the data points lie on three curves in the input space so that they cannot be linearly represented with each other directly. After a nonlinear transformation, these data points lie on three lines in the hidden space and can be linearly represented by each other. The proposed method consists of the following four steps: 1) projecting the data from the input space into a hidden space in which the mapped data lie on linear subspaces; 2) calculating the global self-expression of the whole data set in the hidden space; 3) eliminating the impact of noise, such as Gaussian noise, by zeroing trivial coefficients; 4) constructing a Laplacian graph using the

obtained coefficients and solving a generalised Eigen-decomposition problem to obtain the clustering membership with the algorithm of k-means.

The main contributions and novelty of this work can be summarised as follows:

- We propose a new robust subspace clustering method, which can cluster the data points drawn from multiple nonlinear subspaces. By exploiting the kernel technique to transform the input samples into the hidden space, we effectively tracked the challenge that TRR cannot handle the data lie on nonlinear subspaces.
- A closed-form solution to KTRR is provided. The solution or the proposed model is a function of the kernel matrix and not directly related to the input data. It makes our method very efficient, especially when handling the problem that has high-dimensional input data.
- The proposed KTRR is further extended to handle large-scale datasets. We transform the scalability issue in KTRR as a kind of out-of-sample problem, and solve it with a "sampling, clustering, and classifying" strategy.
- We apply the KTRR method for several image clustering problems. Extensive experiments are conducted to investigate the effectiveness and efficiency of KTRR. The empirical results show that our method significantly outperforms current state-of-the-art subspace clustering algorithms in terms of accuracy, robustness, and computational cost.

The rest of this paper is organised as follows. Section 2 reviews the related work. Section 3 is devoted to the formulation of kernel truncated regression and the presentation of the proposed method. Section 4 reports the experimental results to evaluate the effectiveness and efficiency of the proposed method. Section 5 concludes the paper.

Notations: In this paper, unless specified otherwise, **lower-case bold letters** represent column vectors, **upper-case bold letters** represent matrices,

and the entries of matrices are denoted with subscripts. For instance, $\mathbf{v}$ is a column vector, v_i is its ith entry. $\mathbf{M}$ is a matrix, m_{ij} is the entry in the ith row, jth column, and $\mathbf{m}_j$ denotes the jth column of $\mathbf{M}$. Moreover, $\mathbf{M}^T$ represents the transpose of $\mathbf{M}$, $\mathbf{M}^{-1}$ denotes the inverse matrix of $\mathbf{M}$, and $\mathbf{I}$ stands for the identity matrix. Table 1 summarises some notations used throughout the paper.

Table 1: Some notations used in this paper

Notation	Definition
m	dimensionality of input data points
n	number of input data points
L	number of underlying subspaces
$\mathbf{X} \in \mathbb{R}^{m \times n}$	input data matrix
$\mathbf{x}_i \in \mathbb{R}^m$	vector of the ith column of $\mathbf{X}$
$\mathbf{X}_i \in \mathbb{R}^{m \times n}$	dictionary matrix for the data point $\mathbf{x}_i$
$\kappa(\mathbf{x}_i, \mathbf{x}_j)$	kernel function
$\mathbf{K} \in \mathbb{R}^{n \times n}$	kernel matrix of the input data matrix
ϕ: $\mathbb{R}^m \to \mathcal{H}$	mapping function from the input space to the kernel space
$\phi(\mathbf{x}_i)$	representation of $\mathbf{x}_i$ in the kernel space
$\mathbf{c}_i \in \mathbb{R}^n$	representation coefficient vector for $\phi(\mathbf{x}_i)$
$\mathbf{C} \in \mathbb{R}^{n \times n}$	linear representation coefficient matrix
$\mathbf{W} \in \mathbb{R}^{n \times n}$	similarity matrix of input data matrix
$\mathbf{L} \in \mathbb{R}^{n \times n}$	normalized Laplacian matrix

2. Related Work

In the past decades, a large number of spectral clustering-based methods have been proposed to achieve subspace clustering in many applications, such as image clustering, motion segmentation and gene expression analysis [7]. The key of these methods is to obtain a block-diagonal similarity matrix whose nonzero

elements are only located on the connections of the points from the same subspace. There are two common strategies to compute the similarity matrix, *i.e.*, pairwise distance-based strategy and linear representation-based strategy. Pairwise distance-based strategy computes the similarity between two points according to their pairwise relationship. For example, the original spectral clustering method adopts the Euclidean distance with Heat Kernel to calculate the similarity as

$$s(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}, \tag{1}$$

where $s(\mathbf{x}_i, \mathbf{x}_j)$ denotes the similarity between the data points $\mathbf{x}_i$ and $\mathbf{x}_j$, and the parameter σ controls the width of the neighborhoods.

Alternatively, linear representation-based approaches assume that each data point can be represented by a linear combination of some other points from the intra-subspace [7, 47, 46]. Based on this assumption, the linear representation coefficient is used as a measurement of similarity. The linear representation-based approaches have achieved promising performance in subspace clustering [7, 22, 32, 23] since they encode the global structure of the data set into the representation coefficient matrix.

By given a data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n] \in R^{m \times n}$, the representation-based methods linearly represent $\mathbf{X}$ and obtain the coefficient matrix $\mathbf{C} \in R^{n \times n}$ in a self-expression manner by solving

$$\min \Re(\mathbf{C}) \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C}, \text{diag}(\mathbf{C}) = \mathbf{0}, \tag{2}$$

where $\text{diag}(\mathbf{C}) = \mathbf{0}$ avoids the trivial solution that uses the data point to represent itself. $\Re(\mathbf{C})$ denotes the adopted prior structured regularisation on $\mathbf{C}$, and the major difference among most existing subspace clustering methods is the choice of $\Re(\mathbf{C})$. For example, SSC [7] enforces the sparsity on the column vectors of $\mathbf{C}$ by adopting ℓ_1-norm via $\Re(\mathbf{C}) = \sum_{i=1}^{n} \|\mathbf{c}_i\|_1$, LRR [22] obtains low rankness by using nuclear-norm with $\Re(\mathbf{C}) = \|\mathbf{C}\|_*$. To further achieve robustness, (2) is extended as follows:

$$\min \Re(\mathbf{C}) + \wp(\mathbf{E}) \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E}, \text{diag}(\mathbf{C}) = \mathbf{0}, \tag{3}$$

where $\mathbf{E}$ stands for the errors induced by the noise and corruption, $\wp(\mathbf{E})$ measures the impact of the errors. Generally, $\wp(\cdot) = \|\mathbf{E}\|_F$ and $\wp(\cdot) = \|\mathbf{E}\|_1$ are used to handle the Gaussian noise and the Laplacian noise, respectively, and $\|\cdot\|_F$ denotes the Frobenius-norm of a matrix.

Due to the assumption of linear reconstruction, those methods failed to achieve nonlinear subspaces clustering. To address this challenging issue, some research have conducted few attempts [28, 43]. However, these methods are computationally inefficient since they need to solve ℓ_1- or nuclear-norm minimisation problems. Some deep learning-based methods [49, 14, 29] are also developed to learn nonlinear relationships. They usually need a large amount of data available to explore the nonlinear relationships of the data and cost a lot on computational resources. To address these issues, we propose a new nonlinear subspace clustering method by integrating the kernel technique into linear representation.

3. The proposed subspace clustering method

This section presents the details of our proposed method. Firstly, we introduce the formulation and the optimisation procedure of KTRR. Then, we illustrate how to use the representation coefficient matrix of KTRR to achieve robust subspace clustering. Next, the computational complexity of the proposed method is analysed. At last, an extension of KTRR is provided for handling large-scale datasets.

3.1. Kernel Truncated Regression Representation

For a given data set $\{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^m$, we define a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n]$. Let ϕ: $\mathbb{R}^m \to \mathcal{H}$ be a nonlinear mapping which transforms the input data into a kernel space $\mathcal{H}$, and $\phi(\mathbf{X}_i) = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_{i-1}), \mathbf{0}, \phi(\mathbf{x}_{i+1}), \dots, \phi(\mathbf{x}_n)]$. After mapping $\mathbf{X}$ into a kernel space, the corresponding $\{\phi(\mathbf{x}_i)\}_{i=1}^n$ is generally believed lying in multiple linear subspaces if an appropriate transforming function is selected [28, 43]. Based on this basic idea, we propose to formulate the

objective function of our KTRR as follows:

$$\min_{\mathbf{c}_i} \frac{1}{2}\|\phi(\mathbf{x}_i) - \phi(\mathbf{X}_i)\mathbf{c}_i\|_2^2 + \frac{\lambda}{2}\|\mathbf{c}_i\|_2^2, \tag{4}$$

where the first term is the reconstruction error in the kernel space, the second term serves as an ℓ_2-norm regularization, and λ is a positive real number, which controls the strength of the ℓ_2-norm regularization term.

For each transformed data representation $\phi(\mathbf{x}_i)$, solving the optimisation problem (4), it gives that

$$\mathbf{c}_i^* = \left(\phi(\mathbf{X}_i)^T\phi(\mathbf{X}_i) + \lambda\mathbf{I}\right)^{-1}\phi(\mathbf{X}_i)^T\phi(\mathbf{x}_i). \tag{5}$$

One can find that the solution in (5) does not require $\phi(\mathbf{x}_i)$ to be explicitly computed. It only needs the dot products of the images of the data in the hidden space. The dot products can sometimes be calculated more efficiently as a direct function of the input data points, without explicitly performing the mapping ϕ. In other words, the computation of the images in the hidden space can be by-passed. A function that performs this direct computation is a kernel function. For some choices of a kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j)$: $\mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$, [38] has shown that κ can obtain the dot product in the kernel space $\mathcal{H}$ induced by the mapping ϕ.

For example, for a set of input data points $\{\mathbf{x}_i\}_{i=1}^n$ lie in a two-dimensional space, we project these points into another space with the mapping function

$$\phi : \mathbf{x}_i = (x_{1i}, x_{2i})^T \mapsto \phi(\mathbf{x}_i) = (x_{1i}^2, x_{2i}^2, \sqrt{2}x_{1i}x_{2i})^T. \tag{6}$$

The mapping function projects the data points from a two-dimensional space to a three-dimensional space where the dot product of two images can be computed as

$$\begin{aligned}\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) &= (x_{1i}^2, x_{2i}^2, \sqrt{2}x_{1i}x_{2i})(x_{1j}^2, x_{2j}^2, \sqrt{2}x_{1j}x_{2j})^T \\ &= x_{1i}^2x_{1j}^2 + x_{2i}^2x_{2j}^2 + 2x_{1i}x_{2i}x_{1j}x_{2j} \\ &= (x_{1i}x_{1j} + x_{2i}x_{2j})^2 = (\mathbf{x}_i^T\mathbf{x}_j)^2.\end{aligned} \tag{7}$$

Therefore, the function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$ is the kernel which can be used to compute the dot products of the images corresponding to the above mapping function ϕ in (8) without explicitly calculating the coordinates of their images. Furthermore, the same kernel computes the dot product corresponding to the following mapping function

$$\phi : \mathbf{x}_i = (x_{1i}, x_{2i})^T \mapsto \phi(\mathbf{x}_i) = (x_{1i}^2, x_{2i}^2, x_{1i}x_{2i}, x_{2i}x_{1i})^T, \tag{8}$$

which demonstrates that the same kernel function may corresponding to several different mapping functions [38]. In return, the mapping function is uniquely corresponding to one kernel function. It helps us select a suitable kernel function for the data being processed. Also, using the kernel technique to compute the dot product of images has lower computational complexity by comparing with computing the dot product with the explicitly mapping process. At last, there is no need to change the formulation of our method in (4) to accommodate the particular choice of kernel function. We can select any suitable kernel for the data set being considered.

It is notable that it still requires $O(n^4 + mn^2)$ to obtain the solution in (5) for the problem of n data points with dimensionality of m. To solve the problem in (4) more efficiently, we firstly rewrite it as

$$\min_{\mathbf{c}_i} \frac{1}{2}\|\phi(\mathbf{x}_i) - \phi(\mathbf{X})\mathbf{c}_i\|_2^2 + \frac{\lambda}{2}\|\mathbf{c}_i\|_2^2, \quad \text{s.t.} \quad \mathbf{e}_i^T \mathbf{c}_i = 0, \tag{9}$$

where $\phi(\mathbf{X}) = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \dots \phi(\mathbf{x}_n)]$, $\mathbf{e}_i$ is a column vector with all zero elements except the ith element is one, and the constraint $\mathbf{e}_i^T \mathbf{c}_i = 0$ eliminates the trivial solution of representing a transformed point by itself.

Using the Lagrangian method, we obtain that

$$L(\mathbf{c}_i) = \frac{1}{2}\|\phi(\mathbf{x}_i) - \phi(\mathbf{X})\mathbf{c}_i\|_2^2 + \frac{\lambda}{2}\|\mathbf{c}_i\|_2^2 + \theta \mathbf{e}_i^T \mathbf{c}_i, \tag{10}$$

where θ is the Lagrangian multiplier. Clearly,

$$\frac{\partial L(\mathbf{c}_i)}{\partial \mathbf{c}_i} = (\phi(\mathbf{X})^T \phi(\mathbf{X}) + \lambda \mathbf{I})\mathbf{c}_i - \phi(\mathbf{X})^T \phi(\mathbf{x}_i) + \theta \mathbf{e}_i. \tag{11}$$

Let $\frac{\partial L(\mathbf{c}_i)}{\partial \mathbf{c}_i} = 0$, we have

$$\mathbf{c}_i^* = (\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda\mathbf{I})^{-1}(\phi(\mathbf{X})^T\phi(\mathbf{x}_i) - \theta\mathbf{e}_i). \tag{12}$$

Multiplying $\mathbf{e}_i^T$ on both sides of (12), and since $\mathbf{e}_i^T\mathbf{c}_i = 0$, it holds that

$$\theta = \frac{\mathbf{e}_i^T(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda\mathbf{I})^{-1}\phi(\mathbf{X})^T\phi(\mathbf{x}_i)}{\mathbf{e}_i^T(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda\mathbf{I})^{-1}\mathbf{e}_i}. \tag{13}$$

Substituting (13) into (12), the optimal solution is given as

$$\mathbf{c}_i^* = \mathbf{q}_i - \mathbf{P}\frac{\mathbf{e}_i^T\mathbf{q}_i\mathbf{e}_i}{\mathbf{e}_i^T\mathbf{P}\mathbf{e}_i}, \tag{14}$$

where $\mathbf{q}_i = \mathbf{P}(\phi(\mathbf{X})^T\phi(\mathbf{x}_i))$, and $\mathbf{P} = (\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda\mathbf{I})^{-1}$.

We can combine all the dot products as a matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ whose elements are calculated as

$$K_{ij} = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = [\phi(\mathbf{X})^T\phi(\mathbf{X})]_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j), \tag{15}$$

where $\phi(\mathbf{X}) = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots \phi(\mathbf{x}_n)]$. The matrix $\mathbf{K}$ is the kernel matrix, which is a symmetric and positive semidefinite matrix. Accordingly, (14) can be rewritten as

$$\mathbf{c}_i^* = \mathbf{v}_i - \mathbf{U}\frac{\mathbf{e}_i^T\mathbf{v}_i\mathbf{e}_i}{\mathbf{e}_i^T\mathbf{U}\mathbf{e}_i}, \tag{16}$$

where $\mathbf{U} = (\mathbf{K} + \lambda\mathbf{I})^{-1}$, $\mathbf{v}_i = \mathbf{U}\mathbf{k}_i$, and $\mathbf{k}_i$ is the ith column vector of $\mathbf{K}$.

Note that only one pseudo-inverse operation is needed for solving the representation problems of all data points. The computational complexity of calculating the optimal solutions in (16) has decreased to $O(n^3 + mn^2)$ for n data points with m dimensions.

It has been proved that, under certain condition, the coefficients over intra-subspace data points are larger than those over inter-subspace data points [32]. After representing the data set by the kernel matrix via (16), we handle the errors by performing a hard thresholding $\mathcal{T}_\eta(\cdot)$ over $\mathbf{c}_i^*$, where $\mathcal{T}_\eta(\cdot)$ keeps η

largest entries in $\mathbf{c}_i$ and sets other entries as zeros, *i.e.*,

$$\mathcal{T}_\eta(\mathbf{c}_i^*) = [\mathcal{T}_\eta(c_{1i}^*), \mathcal{T}_\eta(c_{2i}^*), \ldots, \mathcal{T}_\eta(c_{ni}^*)]^T \tag{17}$$

and

$$\mathcal{T}_\eta(c_{ji}^*) = \begin{cases} c_{ji}^*, & \text{if} \quad c_{ji}^* \in \Omega_i; \\ 0, & \text{otherwise}, \end{cases} \tag{18}$$

where Ω_i consists of the η largest elements of $\mathbf{c}_i^*$. Typically, the optimal η equals to the dimensionality of corresponding kernel subspace, which can be estimated by subspace learning approaches [30]. In this manner, it avoids to formulate the impact of the noises into the optimisation problem explicitly and does not need prior knowledge about the errors.

3.2. KTRR for Robust Subspace Clustering

In this section, we present the method to achieve subspace clustering by incorporating KTRR into the spectral clustering framework [25].

For a given data set $\mathbf{X}$, which consists of n data points in $\mathbb{R}^m$, we assume that these points lie on a union of L low-dimensional nonlinear subspaces. We propose to transform the data points into a hidden space, in which the images of these data points can be linearly represented by the images of the data points from the intra-subspace. From (14), we find that the calculation of the representation coefficients does not require the transforming function in an explicit form, but the dot products of the images are needed. We can induce a kernel function to calculate these dot products and obtain the representation coefficients via (16).

Moreover, the existence of the noises in the input dataset leads to some error connections among the data points from different subspaces. We propose to remove these errors through hard thresholding on each column vector of the coefficient matrix $\mathbf{C}^*$ via (17).

These representation coefficients can be seen as the similarities among the input data points. The similarity between two intra-subspace data points is large, and the similarity between two inter-subspace data points is zero or is

approximately equal to zero. Therefore, we can build a similarity matrix $\mathbf{W}$ based on the obtained coefficient matrix $\mathbf{C}^*$ as

$$\mathbf{W} = |(\mathbf{C}^*)^T| + |\mathbf{C}^*|. \tag{19}$$

The matrix of $\mathbf{W}$ is symmetric and is suitable for integrating into the spectral clustering framework.

Then, we compute the normalized Laplacian matrix by following [25]:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}, \tag{20}$$

where $\mathbf{D}$ is a diagonal matrix with $d_{ii} = \sum_{j=1}^{n} w_{ij}$. $\mathbf{L}$ is positive semi-definite and has an eigenvalue equals zero with the eigenvector $\mathbf{D}^{\frac{1}{2}}\mathbf{1}$ [41], where $\mathbf{1} = [1, \ldots, 1]^T \in \mathbb{R}^n$.

Next, we calculate the first L eigenvectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L$ of $\mathbf{L}$, which corresponding to its first L smallest nonzero eigenvalues, and construct a matrix $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L] \in \mathbb{R}^{n \times L}$.

Finally, we apply the k-means clustering algorithm to the matrix $\mathbf{Y}$, by treating each row vector of $\mathbf{Y}$ as a data point. In this way, we can cluster the data into L groups and obtain the clustering membership. The proposed subspace clustering algorithm is summarised in Algorithm 1.

3.3. Computational Complexity Analysis

Given a data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, KTRR takes $O(mn^2)$ to compute the kernel matrix $\mathbf{K}$. Then it takes $O(n^3)$ to obtain the matrix $\mathbf{U}$, and $O(mn^2)$ to calculate all the solutions in (16) with the matrices $\mathbf{U}$ and $\mathbf{K}$. Finally, it requires $O(\eta \log \eta)$ to find η largest coefficients in each column of the representation matrix $\mathbf{C}^*$. Putting these steps together, we obtain the computational complexity of KTRR as $O(mn^2 + n^3)$. This computational complexity is the same as that of TRR, and is considerably less than that of KSSC ($O(mn^2 + tn^3)$)[28], KLRR($O(t(r_{\mathbf{X}} + r)n^2)$)[43], where t denotes the total number of iterations for the corresponding algorithm, $r_{\mathbf{X}}$ is the rank of $\mathbf{X}$, and r is the rank for partial SVD at each iteration of KLRR.

Algorithm 1 Learning kernel truncated regression representation for robust subspace clustering

Input: A given data set $\mathbf{X} \in \mathbb{R}^{m \times n}$, the tradeoff parameter λ, the parameter η, and the number of subspaces L.

Output: The clustering labels of the input data points.

1: Calculate the kernel matrix $\mathbf{K}$ and the matrix $\mathbf{U}$ in (16) and store them.
2: For each point $\mathbf{x}_i \in \mathbb{R}^m$, calculate its linear representation coefficients in the kernel space $\mathbf{c}_i^* \in \mathbb{R}^n$ via (16).
3: Remove trivial coefficients from $\mathbf{c}_i^*$ by performing hard thresholding $\mathcal{T}_\eta(\mathbf{c}_i^*)$, *i.e.*, keeping η largest entries in $\mathbf{c}_i^*$ and zeroing all other elements.
4: Construct a symmetric similarity matrix via (19).
5: Calculate the normalised Laplacian matrix $\mathbf{L}$ via (20).
6: Compute the eigenvector matrix $\mathbf{Y} \in \mathbb{R}^{n \times L}$ that consists of the first L normalised eigenvectors of $\mathbf{L}$, corresponding to its L smallest nonzero eigenvalues.
7: Apply the k-means algorithm to cluster the rows of $\mathbf{Y}$ into L groups and obtain the clustering membership.

3.4. Handling Large-Scale Data Sets with KTRR

From the above computational complexity analysis, we can see that even KTRR is much faster than many existing subspace clustering methods, *e.g.*, KLRR, KSSC, LRR, SSC. However, it is still unable to handle large-scale datasets. Inspired by the strategy in [31], we extend KTRR to handle large-scale datasets with the following three steps: 1) sampling, 2) clustering, and 3) classification.

In the first step, it assumes that the sampled data subset and the whole data set are independent and identically distributed (*i.e.*, i.i.d.) so that the out-of-sample data could be represented by the sampled data. This assumption is general on which most of machine learning algorithms are based. In this paper, we adopt a uniform random sampling approach to sample a subset of data from the input data set. Then, we use the proposed KTRR to obtain the clustering

membership of the sampled data. In the third step, we train a classifier with the sampled data and the corresponding labels and classify the out-of-sample data with the trained classifier.

For the classification of out-of-sample data, the adopted algorithm should be capable of handling data that lie on nonlinear subspaces. We propose to take a fully-connected three-layer (10 hidden units) feed-forward neural network to achieve the classification task since it is easy to train and is potentially able to handle data which lie on nonlinear subspaces. Some other deep neural networks can also be used in this framework. A potential problem of adopting a deep neural network is that when the dimension of input data is very high, a small number of in-sample data may not be sufficient for learning the parameters of the neural network. The procedure of the extended KTRR (EKTRR) is summarised in Algorithm 2.

Algorithm 2 Clustering large-scale datasets with EKTRR

Input: A given data set $\mathbf{X} \in \mathbb{R}^{m \times n}$, the tradeoff parameter λ, thresholding parameter η, the number of sampled data points δ, and the number of subspaces L.

Output: The clustering labels of the input data points.

1: Randomly select δ data columns from $\mathbf{X}$ as in-sample data matrix $\mathbf{A}$. The remaining samples are denoted as out-of-sample data $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{n-\delta})$.
2: Perform KTRR (Algorithm 1) on $\mathbf{A}$ to obtain the cluster membership of $\mathbf{A}$.
3: Train a feed-forward neural network with $\mathbf{A}$ and the corresponding labels.
4: Classify each out-of-sample data point in $\mathbf{Z}$ and obtain the cluster membership of $\mathbf{Z}$.

4. Experimental Study

In this section, we experimentally evaluate the performance of the proposed method. We consider the results in terms of three aspects: 1) accuracy, 2) robustness, and 3) computational cost. Robustness is evaluated by conducting

experiments using samples with two different types of corruptions, *i.e.*, Gaussian noises and random pixel corruption.

4.1. Databases

Six popular image databases are used in our experiments, including Extended Yale Database B (ExYaleB) [19], Columbia Object Image Library (COIL20) [36], Columbia Object Image Library (COIL100) [35], USPS [13], MNIST [18], and Covtype [1]. We give the details of these databases as follows:

- The ExYaleB database contains $2,414$ frontal face images of 38 subjects and around 64 near frontal images under different illuminations per individual, where each image is manually cropped and normalised to the size of 32×32 pixels [2].
- The COIL20 and COIL100 databases contain 20 and 100 objects, respectively. The images of each object were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images. The size of each image is 32×32 pixels, with 256 grey levels per pixel [2].
- The USPS handwritten digit database[1] includes ten classes ($0-9$ digit characters) and 11,000 samples in total. We use a popular subset contains $9,298$ handwritten digit images for the experiments, and all of these images are normalised to the size of 16×16 pixels. In the experiment, we select 200 samples of each subject from the database randomly by following the strategy in [4].
- The MNIST handwritten digit database includes ten classes ($0-9$ digit characters) and $60,000$ samples in total. We use first $10,000$ handwritten digit images of the training subset to conduct the experiments, and all of these images are normalised to the size of 28×28 pixels. In the experiment,

[1]The USPS database and MNIST database used in this paper are download from `http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html`.

we also select 200 samples of each subject from the database randomly to evaluate the performance of different algorithms.

- The Covtype database is a large-scale database, which consists of $581,012$ samples of 7 subjects, and each sample has 54 attribute variables. This database is developed to investigate the performance of algorithms on predicting forest cover type from cartographic variables only (no remotely sensed data). Independent variables were derived from data originally obtained from the US Geological Survey (USGS) and USFS data. In the experiments, we normalise the attribute values to the interval $[0, 1]$.

4.2. Baselines and Evaluation Metrics

We compare KTRR[2] with the state-of-the-art subspace clustering approaches, including truncated regression representation (TRR) [32], two versions of least squares regression (LSR1, LSR2) [24], kernel low-rank representation (KLRR) [43], kernel sparse subspace clustering (KSSC) [28], Latent low-rank representation (LatLRR) [23], low-rank representation with ℓ_1-norm (LRR1) [22], low-rank representation with ℓ_{21}-norm (LRR2) [22], sparse subspace clustering (SSC) [7], sparse manifold clustering and embedding (SMCE) [6], local subspace analysis (LSA) [45], and standard spectral clustering (SC) [25], on four real-world databases. To evaluate the performance of the extension of KTRR (EKTRR) on large-scale set, we compare it with seven scalable clustering algorithms (SSSC [33], SLRR [31], SLSR [31], KASP [44], Nyström [3], SEC [26], and AKK [5]). Furthermore, we also compare the proposed method with currently developed deep subspace clustering methods including deep adversarial subspace clustering (DASC) [49], deep subspace clustering network with ℓ_2-norm regularization on (DSC-Net-L2) [14], deep subspace clustering network with ℓ_1-norm regularization on (DSC-Net-L1) [14], structured autoencoders (StructAE) [29], and SSC with pre-trained convolutional autoencoder features (AE+SSC).

[2]The source code is available at `https://liangli-zhen.github.io/code/KTRR.zip`.

Four popular metrics are adopted to evaluate the subspace clustering quality, *i.e.*, accuracy (AC) [4], normalized mutual information (NMI) [4], the adjusted rand index (ARI) [12], and F-Score [10]. The values of these four metrics are higher if the method works better. The values of these four metrics are equal to 1 indicates the predict result is perfectly matching with the ground truth, whereas 0 indicates totally mismatch.

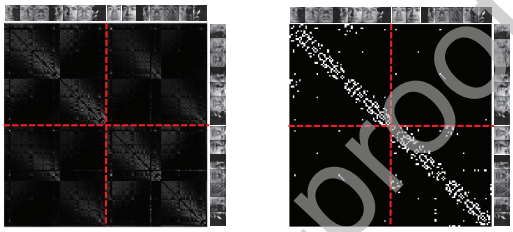
4.3. Visualisation of Representation and Similarity Matrices

Before evaluating the clustering performance of the proposed method, we demonstrate the visualisation results of the coefficient matrix of KTRR with the Gaussian kernel and the obtained similarity matrix. We conduct the experiment on the first 128 facial images of the ExYaleB database, in which the first 64 samples of which belong to the first subject, and the other 64 samples belong to the second subject. We set the parameters as $\lambda = 5$ and $\eta = 4$. The representation matrix $\mathbf{C}^*$ in (16) and the constructed similarity matrix $\mathbf{W}$ in (19) are shown in Fig. 2(a) and Fig. 2(b), respectively.

From Fig. 2(a), we can see that most of the non-zero elements are located in the upper-left part and the bottom-right part, but there still exist some non-zero elements in the upper-right part and the bottom-left part. That is to say, the connections among the same subject are much stronger than that among different subjects, while there still exist numbers of trivial connections among the samples from different subjects since the samples from these subjects are all facial images, which have some common characteristics.

It is well known that an ideal similarity matrix for the spectral clustering algorithm is a block diagonal matrix, *i.e.*, the connections should only exist among the data points from the same cluster [22, 32, 28, 43], such that a hard thresholding operation has been executed. From the result of the similarity matrix $\mathbf{W}$ (in Fig. 2(b)), we find that:

- Most of the bright spots lie in the diagonal blocks of the similarity matrix, *i.e.*, the strong connections exist among the samples from the same subject;



(a) (b)

Figure 2: The visualisation of the representation matrix and the similarity matrix on 128 facial images. They belong to the first 2 subjects in ExYaleB database. (a) The representation matrix in (16). (b) The similarity matrix obtained by our algorithm. The experiment was carried out on the first two subjects of ExYaleB. The top rows and the right columns illustrate some images of these two subjects. The dotted lines split each matrix into four parts. The upper-left part: the similarity relationship among the 64 images of the first subject. The bottom-right part: the similarity relationship among the 64 images of the second subject. The upper-right part and the bottom-left part: the similarity relationship among the images from different subjects. From the connection results, it is easy to find that most of the non-zero elements are located in the upper-left part and the bottom-right part, which means that our method reflects the relationships among the samples from different subjects very well.

- Our method reveals the latent structure of data that these images belong to two subjects. There exist only a few bright spots in the upper-right part and the bottom-left part of the obtained similarity matrix, *i.e.*, the trivial connections among the samples from different subjects have been mostly removed by using the thresholding processing;

- The obtained similarity matrix is a symmetric matrix, which can be directly used for subspace clustering under the framework of the spectral clustering [25].

4.4. Different Kernel Functions

There are many kernel functions, and some of them are commonly used, *e.g.*, polynomial kernels, radial basis functions, and sigmoid kernels. To investigate the performance of the proposed method using different kernels, we study six different kernel functions. We conduct the experiments on four databases, *i.e.*, ExYaleB, COIL20, USPS and MINIST. The clustering performance with different kernels are shown in Table 2, from which we have the following observations:

- The kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}}$ achieves the best performance on ExYaleB and USPS, and obtains competitive performance compared with the kernel functions $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$ and $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma}}$ on COIL20 and MNIST, respectively.

- The performance of the proposed method with the kernel functions $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^3$ and $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$ on ExYaleB is poor. It illustrates that the facial images cannot be project into linear spaces with the mapping function corresponding to these polynomial kernel functions.

- The performance of the proposed method with $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$ outperforms $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^3$ on COIL20, which is different from that on USPS. The same observation can be obtained on results with $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma}}$ and $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}}$. It is mainly caused by the fact that the images from USPS lie in much higher nonlinear subspaces than those

Table 2: Clustering performance (mean ± sd, %) comparison of different kernel functions used in the proposed method with 10 runs, in each run the k-means clustering step is repeated 500 times. The parameter of σ is setting as the mean of the distances between all the samples. The best mean results in different metrics are in bold.

Function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$		$(\mathbf{x}_i^T \mathbf{x}_j)^3$	$(\mathbf{x}_i^T \times_j)^2$	$e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}}$	$e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma}}$	$\frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|^2}$	$\frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|}$
ExYaleB	AC	10.62 ± 0.58	65.02 ± 3.22	**84.82 ± 5.75**	77.43 ± 3.70	78.84 ± 3.90	76.39 ± 3.76
	NMI	14.03 ± 0.87	74.85 ± 1.59	**89.52 ± 2.43**	81.07 ± 1.68	86.21 ± 0.74	80.04 ± 1.36
	ARI	1.32 ± 0.17	50.04 ± 5.77	**77.09 ± 5.84**	62.08 ± 4.18	72.92 ± 2.41	58.82 ± 3.88
	F-Score	4.37 ± 0.25	51.47 ± 5.49	**77.72 ± 5.66**	63.15 ± 4.03	73.66 ± 2.34	60.01 ± 3.71
COIL20	AC	70.99 ± 6.79	85.03 ± 0.92	90.25 ± 6.13	**91.81 ± 0.00**	90.56 ± 0.00	**91.81 ± 0.00**
	NMI	81.04 ± 3.00	94.11 ± 0.27	94.71 ± 2.75	**96.24 ± 0.00**	95.23 ± 0.00	96.06 ± 0.00
	ARI	61.55 ± 8.39	82.98 ± 0.07	88.04 ± 5.49	**91.14 ± 0.00**	89.11 ± 0.00	90.89 ± 0.00
	F-Score	63.71 ± 7.71	83.92 ± 0.07	88.65 ± 5.19	**91.60 ± 0.00**	89.66 ± 0.00	91.36 ± 0.00
USPS	AC	80.38 ± 19.04	72.31 ± 18.06	**81.36 ± 14.93**	74.97 ± 12.40	79.62 ± 17.69	73.64 ± 14.46
	NMI	76.08 ± 9.75	67.38 ± 14.41	**78.04 ± 6.63**	75.59 ± 9.22	74.18 ± 13.08	74.58 ± 8.01
	ARI	70.10 ± 15.43	59.48 ± 17.11	**71.73 ± 12.67**	65.98 ± 13.81	68.32 ± 19.86	64.62 ± 14.44
	F-Score	73.25 ± 13.45	63.78 ± 15.06	**74.69 ± 11.13**	69.72 ± 11.83	71.68 ± 17.47	68.53 ± 12.38
MNIST	AC	65.58 ± 11.65	**66.48 ± 14.54**	63.97 ± 8.11	59.21 ± 14.27	61.62 ± 12.43	62.06 ± 9.67
	NMI	64.27 ± 6.25	63.49 ± 6.50	**66.81 ± 4.43**	63.54 ± 12.66	64.00 ± 9.32	65.33 ± 5.59
	ARI	51.62 ± 10.21	51.54 ± 11.54	**52.63 ± 5.04**	48.62 ± 16.10	50.18 ± 11.25	51.20 ± 7.18
	F-Score	56.70 ± 8.92	56.61 ± 10.09	**57.92 ± 4.16**	54.50 ± 13.94	55.63 ± 9.71	56.66 ± 6.13

from the COIL20, and the functions $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^3$ and $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}}$ induced a much more nonlinear mapping than $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$ and $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma}}$, respectively.

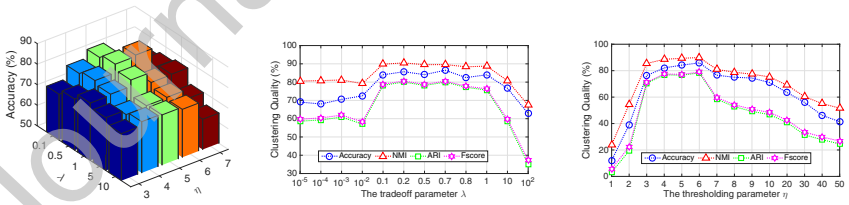
- The selection of different kernels results in a significant difference in the subspace clustering performance both on these four databases.

Since the proposed method with Gaussian kernel can obtain promising performance from the above experimental results, and it is also the most commonly used kernel [38, 43], we adopt the Gaussian kernel function to compute the kernel matrix for our method in the rest of the experiments.

4.5. Parameter Analysis

KTRR has two parameters, the tradeoff parameter λ and the parameter η. The selection of the values of the parameters depends on the data distribution. A bigger λ is suitable for highly corrupted databases, and η corresponds to the dimensionality of the corresponding subspace for the mapped data points.

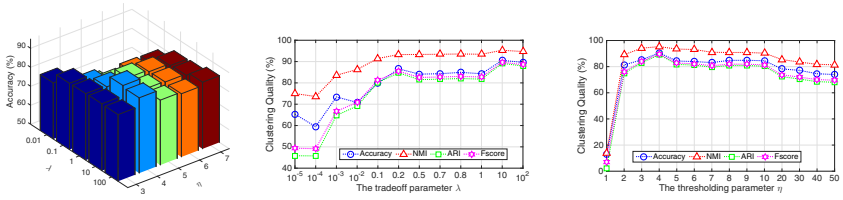
To evaluate the impact of λ and η, we conduct the experiment on the ExYaleB and COIL20 databases. We set the λ from 10^{-5} to 10^2, and η from 1 to 50, the results are shown in Fig. 3 and Fig. 4.



(a) (b) (c)

Figure 3: Clustering performance (mean of 50 runs) of the proposed method on the ExYaleB database. (a) Clustering performance of the proposed method versus different values of λ and η. (b) Clustering performance of the proposed method versus different values of λ under $\eta = 5$. (c) Clustering performance of the proposed method versus different values of η under $\lambda = 0.5$.

From the results, we have the following observations:



(a) (b) (c)

Figure 4: Clustering performance (mean of 50 runs) of the proposed method on the COIL20 database. (a) Clustering performance of the proposed method versus different values of λ and η. (b) Clustering performance of the proposed method versus different values of λ under $\eta = 4$. (c) Clustering performance of the proposed method versus different values of η under $\lambda = 10$.

- KTRR achieves the best clustering performance with λ and η as 0.1 and 5 on the ExYaleB database, and 10 and 4 on the COIL20 database, respectively.

- KTRR can obtain satisfactory performance with λ from 0.1 to 1 on the ExYaleB database, where the values of Accuracy, NMI, ARI, and F-Score are more than 85%, 90%, 75%, and 75%, respectively, and with λ from 0.2 to 100 on the COIL20 database, where the Accuracy, NMI, ARI, and F-Score are more than 80%, 90%, 80%, and 80%. The performance of KTRR is not sensitive to the parameter of λ, which enables KTRR being suitable for the real-world applications.

- The clustering quality with η from 3 to 10 on the ExYaleB and the COIL20 databases are much better than other cases. It means that the thresholding process is helpful to improve the performance of KTRR, and the dimensions of the hidden subspaces of the ExYaleB and the COIL20 databases are in the 3 to 10 range.

4.6. Clustering Performance with Different Number of Subjects

In this subsection, we investigate the clustering performance of the proposed method with a different number of subjects on the COIL100 image database.

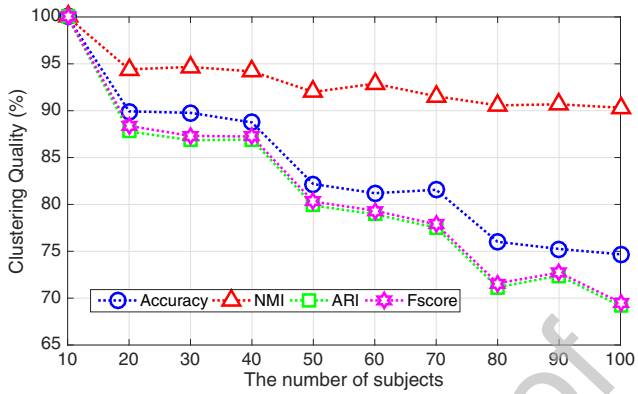


Figure 5: The clustering mean quality of the proposed method on the first t subjects of the COIL100 database with 50 runs.

The experiments are carried out on the first t classes of the database, where t increases from 10 to 100 with an interval of 10. The clustering results are shown in Fig. 5, from which we can see that:

- In general, with the number of subjects increase, the clustering performance is decreased since the clustering difficulty is increasing with the number of subjects.
- With an increasing number of subjects, the NMI of KTRR is changed slightly, varying from 100% to 90%. The potential reason is that the NMI is robust to the data distribution (increasing subject number) [32].
- The proposed method obtains satisfactory performance on the COIL100 database. It achieves perfect clustering result for $t = 10$, and gets the satisfactory performance at $t = 100$ with Accuracy, NMI, ARI, and F-Score be around 74%, 90%, 68%, and 68%, respectively.

4.7. Comparison with Existing Methods on Clean Images

In this experiment, we compare KTRR with other 12 state-of-the-art approaches on four different benchmark databases, *i.e.*, Extended Yale Database B (ExYaleB) [19], Columbia Object Image Library (COIL20) [36], USPS [13],

MNIST [18]. The performance comparisons on COIL100 and Covtype are not provided here since several tested algorithms need a very long period of time to obtain their results on these two databases. For a fair comparison, we use the same spectral clustering framework [25] with different similarity matrices obtained by the tested methods. According to the setting in [43], for all of the kernel-based algorithms, we adopt the commonly-used Gaussian kernel on all datasets and use the default bandwidth parameter which is set to the mean of the distances between all the samples. For each dataset, we perform each method 10 runs, in each run the k-means clustering step is repeated 500 times and report the mean and the standard deviation of the used metrics. The clustering results on the above four databases are shown in Table 3 - Table 6. The best means for each database are highlighted in boldface. To have statistically sound conclusions, the Wilcoxon's rank sum test [9] at a 0.05 significance level is adopted to test the significance of the differences between the results obtained by the proposed method and all other algorithms. From the results, we can obtain the following conclusions.

Table 3: Clustering performance (mean ± sd, %) comparisons of different methods on the ExYaleB database. The best mean results in different metrics are in bold. The "†" indicates that the value of the proposed method is significantly different from all other methods at a 0.05 level by the Wilcoxon's rank sum test.

Methods	AC	NMI	ARI	F-Score
KTRR	**84.82±5.75†**	**89.52±2.43†**	**77.09±5.84†**	**77.72±5.66†**
TRR	67.04±2.93	72.20±2.61	41.07±6.91	43.01±6.52
LSR1	55.56±3.36	58.59±1.45	33.24±2.15	35.20±2.02
LSR2	51.07±4.45	54.03±2.59	25.42±2.41	27.78±2.24
KLRR	52.30±4.31	61.98±2.45	36.06±3.49	37.87±3.37
KSSC	58.41±3.19	64.41±1.10	32.40±5.82	34.59±5.38
LatLRR	51.40±3.36	54.41±1.76	27.10±2.26	29.33±2.07
LRR1	50.32±2.68	53.31±1.42	26.42±2.17	28.66±2.00
LRR2	49.80±4.72	53.26±2.22	25.63±2.70	27.93±2.52
SSC	52.87±5.46	58.02±3.44	24.20±4.74	26.83±4.34
SMCE	48.91±3.71	60.22±1.28	30.46±3.06	32.54±2.84
LSA	33.97±3.95	47.38±1.87	20.98±1.45	23.20±1.36
SC	19.69±1.70	32.96±1.54	10.16±1.01	12.56±0.98

(1) Evaluation on the ExYaleB facial database

- The KTRR algorithm achieves the best results in the tests and gains a significant improvement over TRR. The means of Accuracy, NMI, ARI, and F-Score of KTRR are about 17%, 17%, 26% and 24% higher than that of TRR, 32%, 28%, 41%, and 40% higher than that of KLRR.

- TRR [32] outperforms LSR1 [24] and LSR2 [24] with a considerable gap. It means that the hard thresholding operator over the coefficient vectors has a significant impact on the performance of TRR since the main difference between them is that the former one has the hard thresholding step.

- All representation-based methods, *i.e.*, KTRR, TRR [32], KLRR [43], KSSC [28], LRR [22] and SSC [7], outperform the standard spectral clustering method [25]. SC is failed due to some parts of the images from different subjects are similar in ExYaleB.

- All the linear representation methods, *i.e.*, TRR [32], LRR [22] and SSC [7], are inferior to their kernel-based extensions, *i.e.*, KTRR, KLRR [43], and KSSC [28]. It means that the nonlinear representation methods are more suitable to model the ExYaleB facial images.

(2) Evaluation on the COIL20 database

- The KTRR algorithm obtains the Accuracy of 90.25%, which is better than all other tested methods. Specifically, the Accuracy of KTRR is about 6.13% higher than that of the second best method TRR, and 10.96% higher than that of the third best method KSSC.

- All the linear representation methods, *i.e.*, TRR [32], LRR [22], and SSC [7], are still inferior to their kernel-based extensions, *i.e.*, KTRR, KLRR [43], and KSSC [28]. Their non-linear versions obtain the Accuracy improvements of 6.13%, 0.68%, and 10%, respectively. It means the images in COIL20 still lie in multiple nonlinear subspaces.

Table 4: Clustering performance (mean ± sd, %) comparisons of different methods on the COIL20 database. The best mean results in different metrics are in bold. The "†" indicates that the value of the proposed method is significantly different from all other methods at a 0.05 level by the Wilcoxon's rank sum test.

Methods	AC	NMI	ARI	F-Score
KTRR	**90.25±6.13†**	**94.71±2.75†**	**88.04±5.49†**	**88.65±5.19†**
TRR	84.12 ± 3.35	91.79 ± 0.94	80.72 ± 3.05	81.76 ± 2.80
LSR1	67.93±0.26	76.98±0.40	59.95±0.41	61.96±0.39
LSR2	67.24±0.30	75.75±0.23	58.53±0.17	60.62± 0.16
KLRR	68.66 ± 5.51	77.93 ± 3.24	61.96 ± 6.98	63.92 ± 6.55
KSSC	79.39 ± 8.15	89.50 ± 2.73	76.54 ± 7.13	77.81 ± 6.65
LatLRR	67.97 ± 3.47	76.78 ± 1.32	60.03 ± 2.42	62.03 ± 2.30
LRR1	67.94 ± 7.97	76.45 ± 2.20	60.03 ± 4.94	62.09 ± 4.66
LRR2	66.59 ± 3.35	75.33 ± 2.50	58.45 ± 4.21	60.57 ± 3.98
SSC	69.39 ± 5.93	80.61 ± 2.44	62.14 ± 5.18	64.17 ± 4.80
SMCE	76.51 ± 15.98	90.51 ± 5.69	75.20 ± 15.45	76.61 ± 14.41
LSA	72.86 ± 6.67	81.49 ± 3.69	68.13 ± 5.92	69.74 ± 5.57
SC	69.17 ± 3.81	79.43 ± 2.18	63.77 ± 3.88	65.60 ± 3.68

- KLRR, LatLRR and two types of LRR methods are all inferior to the standard spectral method. It means that the mapped data of COIL20 cannot be represented by other mapped data with the low-rank constraint.

(3) Evaluation on the USPS handwriting database

- The KTRR algorithm achieves the best results in the tests. The value of Accuracy of KTRR is about 21% higher than that of the TRR, 10% higher than that of KLRR, and 6% higher than that of KSSC. The performance results of KTRR on NMI, ARI, and F-Score are also higher than other methods.

- All the linear representation methods, *i.e.*, TRR [32], LRR [22], and SSC [7], are inferior to their kernel-based extensions, *i.e.*, KTRR, KLRR [43], and KSSC [28]. The performance improvement is considerable, *e.g.*, the Accuracy value of KSSC is about 44% higher than that of SSC.

- SSC is inferior to LRR, while its kernel-based extension KSSC achieves a good performance. The implicit transformation on the USPS images

Table 5: Clustering performance (mean ± sd, %) comparisons of different methods on the USPS handwriting database. The best mean results in different metrics are in bold. The "†" indicates that the value of the proposed method is significantly different from all other methods at a 0.05 level by the Wilcoxon's rank sum test.

Methods	AC	NMI	ARI	F-Score
KTRR	**81.36 ± 14.93†**	**78.04 ± 6.63†**	**71.73 ± 12.67†**	**74.69 ± 11.13†**
TRR	67.98±14.18	71.72±6.59	59.76±12.36	64.17±10.61
LSR1	61.80±8.59	60.22±7.00	47.42±11.78	52.97±10.43
LSR2	62.54±9.76	61.37±4.38	48.57±8.29	53.99±7.28
KLRR	70.72 ± 2.63	66.23 ± 3.35	57.21 ± 3.76	61.64 ± 3.41
KSSC	75.17 ± 2.89	73.97 ± 2.41	65.11 ± 3.67	68.76 ± 3.28
LatLRR	70.97 ± 4.10	66.55 ± 4.37	57.20 ± 4.51	61.59 ± 4.11
LRR1	70.16 ± 4.29	66.69 ± 4.63	57.18 ± 4.79	61.58 ± 4.34
LRR2	70.91 ± 3.96	66.87 ± 4.35	57.50 ± 4.48	61.85 ± 4.08
SSC	26.86 ± 13.39	20.93 ± 13.44	9.95 ± 13.56	24.07 ± 7.62
SMCE	73.77 ± 7.85	71.29 ± 9.69	62.08 ± 13.10	66.10 ± 11.63
LSA	68.51 ± 8.95	64.80 ± 7.93	55.58 ± 9.00	60.30 ± 7.99
SC	70.79 ± 8.58	62.72 ± 4.55	53.55 ± 5.24	58.25 ± 4.67

makes the mapped data points to be much better represented with each other in a sparse representation form. The performance of SSC is poor on this database. It is mainly because the images in each group are not sufficient, which makes SSC result in a wrong representation for the data and suffer a low clustering accuracy.

(4) Evaluation on the MNIST handwriting database

- The proposed method KTRR achieves the best clustering result and obtains a significant improvement of 31.78% at Accuracy on TRR. The indexes NMI, ARI, and F-Score of KTRR are also higher than all other tested methods.

- All the linear representation methods, *i.e.*, TRR [32], LRR [22], and SSC [7], are inferior to their kernel-based extensions, *i.e.*, KTRR, KLRR [43], and KSSC [28]. Especially, LRR results in poor performance on this database, while its kernel-based version, KLRR, obtains much better clustering quality regarding Accuracy, NMI, ARI, and F-Score. It demon-

Table 6: Clustering performance (mean ± sd, %) comparisons of different methods on the MNIST handwriting database. The best mean results in different metrics are in bold. The "†" indicates that the value of the proposed method is significantly different from all other methods at a 0.05 level by the Wilcoxon's rank sum test.

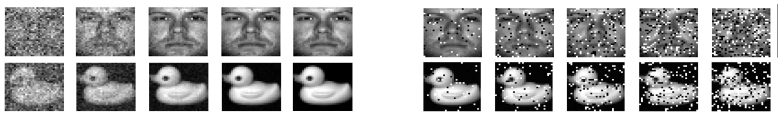
Methods	AC	NMI	ARI	F-Score
KTRR	**63.97 ± 8.11**†	**66.81 ± 4.43**†	**52.63 ± 5.04**†	**57.92 ± 4.16**†
TRR	54.05±6.40	54.79±5.04	39.72±6.66	46.38±5.75
LSR1	50.87±4.90	45.70±4.09	32.92±3.44	39.78±3.12
LSR2	46.77±5.15	43.37±4.20	29.59±4.12	36.90±3.52
KLRR	61.31 ± 6.14	60.07 ± 5.86	47.46 ± 5.77	52.99 ± 4.92
KSSC	57.13 ± 15.57	59.43 ± 13.06	45.02 ± 16.11	50.99 ± 14.05
LatLRR	14.48 ± 8.37	4.04 ± 7.96	1.26 ± 4.47	18.13 ± 2.74
LRR1	18.08 ± 10.57	6.76 ± 11.18	2.80 ± 6.50	18.42 ± 4.13
LRR2	18.52 ± 12.34	7.53 ± 14.97	3.27 ± 8.64	18.43 ± 5.15
SSC	22.02 ± 20.53	14.58 ± 24.25	6.16 ± 16.06	21.67 ± 9.34
SMCE	61.66 ± 5.59	59.08 ± 3.99	46.81 ± 5.80	52.39 ± 5.05
LSA	63.03 ± 7.46	61.94 ± 5.78	49.50 ± 8.19	54.78 ± 7.28
SC	55.11 ± 5.45	48.80 ± 5.30	36.95 ± 5.62	43.38 ± 5.01

strates the advantage of the kernel-based methods when dealing with high-dimensional data.

- KTRR, KLRR, SMCE and LSA achieve the best clustering results on the MNIST handwriting images compared with other methods. However, the performances of all the test methods are not well. A more suitable kernel function may help these methods to get better performance on this type of databases.

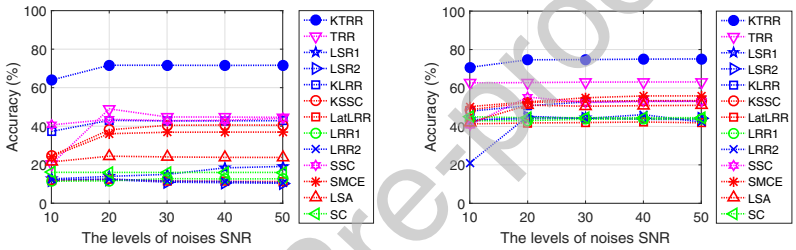
4.8. Comparison with Existing Methods on Corrupted Images

To evaluate the robustness of the proposed method, we conduct the experiments on the first 10 subjects of the COIL20 database and ExYaleB database respectively. All used images are corrupted by additive white Gaussian noises or random pixel corruptions. Some corrupted image samples under different levels of noises are as shown in Fig. 6. For additive Gaussian noises, we add the noises with SNR $= 10, 20, 30, 40, 50dB$; For the random pixel corruptions, we adopt the pepper & salt noises with the ratios of affected pixels being $5\%, 10\%, 15\%, 20\%, 25\%$.



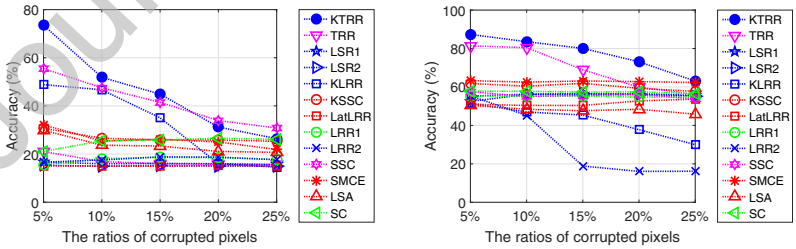
(a) (b)

Figure 6: (a) The corrupted samples with additive Gaussian noises under SNR equals $10, 20, 30, 40, 50$ dB from left to right. (b) The corrupted samples with pepper and salt noises under the ratio of affected pixels equals $5\%, 10\%, 15\%, 20\%, 25\%$ from left to right.



(a) (b)

Figure 7: The clustering accuracy (mean of 50 runs) on images with different levels of additive Gaussian noises. (a) The clustering accuracy on the ExYaleB database. (b) The clustering accuracy on the COIL20 database.



(a) (b)

Figure 8: The clustering accuracy (mean of 50 runs) on images with different ratios of pepper & salt corruptions. (a) The clustering accuracy on the ExYaleB database. (b) The clustering accuracy on the COIL20 database.

The clustering quality of the compared methods on the two databases with Gaussian noises is shown in Fig. 7, from which we can get the following observations:

- The proposed KTRR is considerably more robust than other methods under Gaussian noises. Specifically, KTRR obtains the Accuracy around 80% under SNR equals $10dB$ on COIL20, which is much higher than all other tested algorithms, especially SC, LSA, LRR1, LRR2, LSR1, and LSR2.
- Most of these spectral-based methods are relatively robust against Gaussian noises. While the performance of LRR1, LRR2, and LatLRR are sharply deteriorated on these two databases. The main reason may be that the additional Gaussian noises have destroyed the underlie the low-rank structure of the representation matrix.
- The accuracy of all tested methods on the COIL20 database is higher than that on the ExYaleB database. It is consistent with the result of that on clean images.

The clustering quality of the compared methods on the images with randomly corruptions is shown in Fig. 8, from which we obtain that:

- The KTRR algorithm is robust to the random pixel corruptions on COIL20. It achieves the best results under the ratio of affected pixels equals 5% to 15%. It obtains the Accuracy around 60% under the ratio of affected pixels equals 25% on the COIL20 database, which is a very challenging situation that we can see in Fig. 6. However, the Accuracy of KTRR drops severely with the increase of the ratios of corrupted pixels, and lower than that of SSC under 20% and 25% of corrupted pixels on ExYaleB. The performance of SSC does not drops sharply, because SSC adopts a ℓ_1-norm constraint to handle the salt & pepper corruptions.
- All the investigated methods perform not as well as the case with white Gaussian noises. The result is consistent with a widely-accepted conclu-

sion that non-additive corruptions are more challenging than additive ones in pattern recognition. The pixel values of the images are changed greater under the salt & pepper corruptions than that under the Gaussian noises.

- All of the algorithms perform much better on the COIL20 database than on the ExYaleB database. The values of Accuracy of all algorithms are lower than 40% on the ExYaleB database under 20% and 25% of corrupted pixels. From Fig. 6, we find that most pixel values of the images from the COIL20 database are close to 0 or 255. This leads to some of the corruptions to be useless and weakens the impact to the final clustering results.

4.9. Comparison of Computational Time Cost

Table 7: Computational time (seconds) comparison of different methods on the ExYaleB, COIL20, USPS, and MNIST databases. The t_1 and t_2 denote the time cost on the similarity graph construction process and the time cost on the whole clustering process of each method respectively. The best mean results in different metrics are in bold.

Dataset	ExYaleB		COIL20		USPS		MNIST	
	t1	t2	t1	t2	t1	t2	t1	t2
KTRR	22.96	47.62	6.50	11.66	16.92	27.82	22.56	33.96
TRR	23.71	48.8	6.54	12.75	11.95	22.74	22.43	32.35
LSR1	0.44	91.24	0.19	14.53	0.32	16.29	0.35	17.54
LSR2	0.43	88.42	0.21	14.39	0.16	14.80	0.24	19.09
KLRR	45.82	71.26	16.11	25.55	29.41	39.93	34.20	44.64
KSSC	5512.68	5543.4	1466.12	1472.07	2752.97	2763.19	5742.89	5753.42
LatLRR	772.44	806.43	579.01	584.46	50.05	58.98	246.35	270.19
LRR1	248.94	286.91	430.23	436.59	43.34	51.88	155.70	167.25
LRR2	270.65	311.34	454.87	460.07	49.10	58.90	172.32	186.82
SSC	2301.75	2313.31	121.76	126.88	62.25	98.79	112.13	153.23
SMCE	10.15	**45.18**	5.76	10.12	67.44	76.50	16.78	25.52
LSA	198.48	229.26	61.14	66.01	108.67	120.46	142.71	154.64
SC	**0.33**	124.45	**0.15**	**7.61**	**0.14**	**11.73**	**1.09**	**14.65**

To investigate the efficiency of KTRR, we compare its computational time with that of other 12 approaches on the clean images of four databases. The time costs for building a similarity graph (t_1) and the whole time cost for clustering (t_2) are recorded to evaluate the efficiency of compared methods.

Table 7 shows the time costs of different methods with the parameters which achieve their best results. We can see that:

- KTRR and TRR [32] are much faster than KSSC, SSC, KLRR, and LRR. The results are consistent with the fact that the theocratical computation complexities of KTRR and TRR are much lower than those of KSSC, SSC, KLRR, and LRR methods. The KTRR and TRR algorithms both have analytical solutions, and only one pseudo-inverse operation is required for solving the representation problems of all data points for KTRR and TRR algorithms.

- The standard SC [25] is the fastest since its similarity graph is computed via the pairwise kernel distances among the input samples, and LSR1 and LSR2 also have a low time cost for the similarity graph construction. KSSC [28] is the most time-consuming method. This implies that KSSC cannot be used to handle large-scale databases directly.

- The time cost of the proposed method is very close to that of its linear version TRR. Specifically, TRR is faster than KTRR on the ExYaleB and the USPS databases, while it is slower than KTRR on the MNIST database. They have similar time costs on the COIL20 database. The Wilcoxon's rank sum test [9] at a 0.05 significance level shows there is no significant difference between the time costs of KTRR and TRR on the similarity graph construction and the whole clustering process on the tested four databases.

4.10. Comparison between KTRR and Deep Subspace Clustering Methods

The deep learning methods have achieved great success in numerous recognition tasks. In this experiment, we compare the proposed methods with deep subspace cluster methods. The comparison result[3] on the COIL20 dataset is shown in Fig. 9, from which we can see that KTRR outperforms AE+SSC in

[3]The results of DASC and StructAE are presented by their authors.

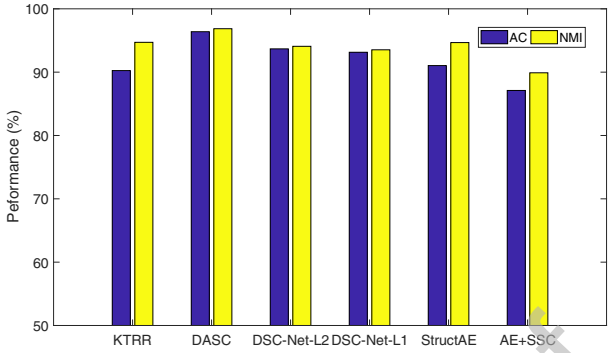


Figure 9: The clustering performance comparison between our method and the deep learning-based clustering methods on the COIL20 dataset.

terms of Accuracy and NMI. Even KTRR is inferior to DASC [49], DSC-Net-L2 [14], DSC-Net-L1 [14] and StructAE [29] in terms of Accuracy, it achieves competitive scores as these four deep subspace clustering methods in terms of NMI. Note that deep learning-based methods perform well only when sufficiently large amounts of data are available. This is a severe limitation in fields in which obtaining such data is either difficult or expensive. KTRR does not need a large number of data samples, and it has a closed-form solution, which costs much less computational resources to obtain the final clustering results.

4.11. Comparison of Performance on Large-Scale Database

To evaluate the capability of handling large-scale data, we compare seven peer algorithms on the Covtype database. Since the cost of computing the values of ARI and F-Score on the large-scale database is extremely high, we only report Accuracy, NMI, and the time cost of the tested algorithms in Table 8. From the results, we have the following observations:

- The extension of KTRR, EKTRR, outperforms the other algorithms. The result of Accuracy of EKTRR is 11.01% higher than the second best algorithm SSSC, and 15.34% higher than the second fastest algorithm Nyström.

- SLSR obtains the highest NMI value, but the NMI values obtained by all the tested algorithms are very small. The metric NMI is not able to distinguish the performance of the tested algorithms in this case.

- The time cost of our method is the lowest. It is at least two times faster than other algorithms. This is due to the fact that our method uses the in-sample data to train the neural network. Then, all other data are classified by the trained network. This makes the proposed method very competitive to handle large-scale databases. Some other

Table 8: Clustering performance (mean ± sd, %) and mean time cost (seconds) comparisons of different methods on the Covtype database (581,012 samples). The best mean results in different metrics are in bold. The "†" indicates that the value of the proposed method is significantly different from all other methods at a 0.05 level by the Wilcoxon's rank sum test.

Methods	AC	NMI	Time
EKTRR	**39.61±5.91**†	5.19± 2.71†	**16.50**†
SSSC [31]	28.60±0.00	5.30 ± 0.00	135.62
SLSR [31]	26.45±0.00	**7.14±0.00**	119.78
SLRR [31]	27.23±0.03	3.65±0.02	122.85
KASP [44]	23.95±1.96	3.55±0.18	913.25
Nyström [3]	24.26±0.61	3.75±0.04	22.95
SEC [26]	21.05±0.01	3.64±0.01	32.05
AKK [5]	22.76±1.65	3.76±0.08	240.65

5. Conclusion

In this paper, we incorporated the kernel technique into the linear representation method to achieve robust nonlinear subspace clustering. It does not need the prior knowledge about the structure of errors in the input data and remedies the drawback of the existing TRR method that it cannot deal with the data points from nonlinear subspaces. Moreover, through the theoretical analysis of our proposed mathematical model, we found that the developed optimisation problem can be solved analytically, and the closed-form solution is only dependent on the kernel matrix. These advantages enable our proposed method being potentially useful in many real-world applications. Comprehensive experiments

on several real-world image databases have demonstrated the effectiveness and efficiency of the proposed method.

In the future, we plan to conduct a systematical investigation on the selection of optimal kernel for our proposed method and study how to determine the number of nonlinear subspaces automatically.

CRediT author statement

Liangli Zhen: Conceptualization, Investigation, Methodology, Software, Formal analysis, Writing - Original draft preparation.

Dezhong Peng: Methodology, Formal analysis, Investigation, Writing - Reviewing and Editing, Supervision, Funding acquisition.

Wei Wang: Validation, Software, Resources, Writing - Reviewing and Editing.

Xin Yao: Methodology, Formal analysis, Visualization, Writing - Reviewing and Editing, Supervision, Funding acquisition.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under grants 61432012, 61329302, the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under grant EP/J017515/1, the Sichuan Science and Technology Planning Projects (Grants No. 2018TJPT0031, 2019YFH0075 and 2018GZDZX0030), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Peacock Plan (Grant No. KQTD2016112514355531), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

References

[1] Blackard, J.A., 1998. Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types. Ph.D. thesis. Fort Collins, CO, USA.

[2] Cai, D., He, X., Han, J., Huang, T.S., 2011. Graph regularized nonnegative matrix factorization for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 1548–1560.

[3] Chen, W.Y., Song, Y., Bai, H., Lin, C.J., Chang, E.Y., 2011. Parallel spectral clustering in distributed systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 568–586.

[4] Cheng, B., Yang, J., Yan, S., Fu, Y., Huang, T.S., 2010. Learning with ℓ^1-graph for image analysis. IEEE Transactions on Image Processing 19, 858–866.

[5] Chitta, R., Jin, R., Havens, T.C., Jain, A.K., 2011. Approximate kernel k-means: Solution to large scale kernel clustering, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, California, USA. pp. 895–903.

[6] Elhamifar, E., Vidal, R., 2011. Sparse manifold clustering and embedding, in: Proceedings of the Advances in Neural Information Processing Systems, Granada, Spain. pp. 55–63.

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

[8] Fan, J., Wang, J., 2018. A two-phase fuzzy clustering algorithm based on neurodynamic optimization with its application for polsar image segmentation. IEEE Transactions on Fuzzy Systems 26, 72–83.

[9] Gibbons, J.D., Chakraborti, S., 2011. Nonparametric statistical inference. Springer.

[10] Goutte, C., Gaussier, E., 2005. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. Springer. pp. 345–359.

[11] He, R., Zhang, Y., Sun, Z., Yin, Q., 2015. Robust subspace clustering with complex noise. IEEE Transactions on Image Processing 24, 4001–4013.

[12] Hubert, L., Arabie, P., 1985. Comparing partitions. Journal of Classification 2, 193–218.

[13] Hull, J.J., 1994. A database for handwritten text recognition research. IEEE Transactions on Pattern Analysis and Machine Intelligence 16, 550–554.

[14] Ji, P., Zhang, T., Li, H., Salzmann, M., Reid, I., 2017. Deep subspace clustering networks, in: Proceedings of the International Conference on Neural Information Processing Systems, USA. pp. 23–32.

[15] Kang, Z., Pan, H., Hoi, S.C., Xu, Z., 2019. Robust graph learning from noisy data. IEEE Transactions on Cybernetics , in press.

[16] Kang, Z., Peng, C., Cheng, Q., 2017. Kernel-driven similarity learning. Neurocomputing 267, 210–219.

[17] Kang, Z., Zhou, W., Zhao, Z., Shao, J., Han, M., Xu, Z., 2020. Large-scale multi-view subspace clustering in linear time, in: Proceedings of the AAAI Conference on Artificial Intelligence, New York, USA.

[18] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of IEEE 86, 2278–2324.

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

[20] Li, Z., Liu, J., Tang, J., Lu, H., 2015. Robust structured subspace learning for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 37, 2085–2098.

[21] Li, Z., Tang, J., He, X., 2017. Robust structured nonnegative matrix factorization for image representation. IEEE Transactions on Neural Networks and Learning Systems 29, 1947–1960.

[22] Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., Ma, Y., 2013. Robust recovery of subspace structures by low-rank representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 35, 171–184.

[23] Liu, G., Yan, S., 2011. Latent low-rank representation for subspace segmentation and feature extraction, in: Proceedings of the International Conference on Computer Vision, Barcelona, Spain. pp. 1615–1622.

[24] Lu, C., Tang, J., Lin, M., Lin, L., Yan, S., Lin, Z., 2013. Correntropy induced L2 graph for robust subspace clustering, in: Proceedings of the International Conference on Computer Vision, Sydney, Australia. pp. 1801–1808.

[25] Ng, A.Y., Jordan, M.I., Weiss, Y., 2002. On spectral clustering: Analysis and an algorithm, in: Proceedings of the Advances in Neural Information Processing Systems, Vancouver, British Columbia, Canada. pp. 849–856.

[26] Nie, F., Zeng, Z., Tsang, I.W., Xu, D., Zhang, C., 2011. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. IEEE Transactions on Neural Networks 22, 1796–1808.

[27] Parsons, L., Haque, E., Liu, H., 2004. Subspace clustering for high dimensional data: a review. SIGKDD Explor. 6, 90–105.

[28] Patel, V.M., Vidal, R., 2014. Kernel sparse subspace clustering, in: Proceedings of the International Conference on Image Processing, IEEE, Paris, France. pp. 2849–2853.

[29] Peng, X., Feng, J., Xiao, S., Yau, W., Zhou, J.T., Yang, S., 2018. Structured autoencoders for subspace clustering. IEEE Transactions on Image Processing 27, 5076–5086.

[30] Peng, X., Lu, J., Yi, Z., Yan, R., 2017a. Automatic subspace learning via principal coefficients embedding. IEEE Transactions on Cybernetics 47, 3583–3596.

[31] Peng, X., Tang, H., Zhang, L., Yi, Z., Xiao, S., 2016. A unified framework for representation-based subspace clustering of out-of-sample and large-scale data. IEEE Transactions on Neural Networks and Learning Systems 27, 2499–2512.

[32] Peng, X., Yu, Z., Yi, Z., Tang, H., 2017b. Constructing the L2-graph for robust subspace learning and subspace clustering. IEEE Transactions on Cybernetics 47, 1053–1066.

[33] Peng, X., Zhang, L., Yi, Z., 2013. Scalable sparse subspace clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 430–437.

[34] Rao, S.R., Tron, R., Vidal, R., Ma, Y., 2008. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Alaska, USA. pp. 1–8.

[35] S. A. Nene, S.K.N., Murase, H., 1996a. Columbia Object Image Library (COIL-100). Report. Columbia University.

[36] S. A. Nene, S.K.N., Murase, H., 1996b. Columbia Object Image Library (COIL-20). Report. Columbia University.

[37] Seung, H.S., Lee, D.D., 2000. The manifold ways of perception. Science 290, 2268–2269.

[38] Shawe-Taylor, J., Cristianini, N., 2004. Kernel methods for pattern analysis. Cambridge Univ. Press.

[39] Soltanolkotabi, M., Elhamifar, E., Cands, E.J., 2014. Robust subspace clustering. The Annals of Statistics 42, 669–699.

[40] Vidal, R., Ma, Y., Sastry, S., 2005. Generalized principal component analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 1945–1959.

[41] Von Luxburg, U., 2007. A tutorial on spectral clustering. Statistics and Computing 17, 395–416.

[42] Wang, Y., Tang, Y.Y., Li, L., 2015. Minimum error entropy based sparse representation for robust subspace clustering. IEEE Transactions on Signal Processing 63, 4010–4021.

[43] Xiao, S., Tan, M., Xu, D., Dong, Z.Y., 2015. Robust kernel low-rank representation. IEEE Transactions on Neural Networks and Learning Systems 27, 2268–2281.

[44] Yan, D., Huang, L., Jordan, M.I., 2009. Fast approximate spectral clustering, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France. pp. 907–916.

[45] Yan, J., Pollefeys, M., 2006. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate, in: Proceedings of the European Conference on Computer Vision, Springer, Graz, Austria. pp. 94–106.

[46] Zhen, L., Li, M., Peng, D., Yao, X., 2020. Objective reduction for visualising many-objective solution sets. Information Sciences 512, 278–294.

[47] Zhen, L., Peng, D., Yi, Z., Xiang, Y., Chen, P., 2017. Underdetermined blind source separation using sparse coding. IEEE Transactions on Neural Networks and Learning Systems 28, 3102–3108.

[48] Zhen, L., Yi, Z., Peng, X., Peng, D., 2014. Locally linear representation for image clustering. Electronics Letters 50, 942–943.

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