Kernelized Sparse Self-Representation for Clustering and Recommendation

Xiao Bian* Feng Li† Xia Ning†

Abstract

Sparse models have demonstrated substantial success in applications for data analysis such as clustering, classification and denoising. However, most of the current work is built upon the assumption that data is distributed in a union of subspaces, whereas limited work has been conducted on nonlinear datasets where data reside in a union of manifolds rather than a union of subspaces. To understand data nonlinearity using sparse models, in this paper, we propose to exploit the self-representation property of nonlinear data in an implicit feature space using kernel methods. We propose a kernelized sparse self-representation model, denoted as KSSR, and a novel Kernelized Fast Iterative Soft-Thresholding Algorithm, denoted as K-FISTA, to recover the underlying nonlinear structure among the data. We evaluate our method for clustering problems on both synthetic and real-world datasets, and demonstrate its superior performance compared to the other state-of-the-art methods. We also apply our method for collaborative filtering in recommender systems, and demonstrate its great potential for novel applications beyond clustering.

Key words: clustering, recommendation, self representation, kernel

1 Introduction

Sparse models on high-dimensional data have drawn keen research interests due to their effectiveness on subspace recovery, denoising, clustering and classification [4, 11, 28, 19]. Typically, the intrinsic degrees of freedom of a given high-dimensional dataset are much less than those allowed by the corresponding ambient high-dimensional space. Discovering the underlying structure hidden from the redundant high dimensions is therefore critical for modeling and analyzing high-dimensional data. A general assumption about the underlying structure of high-dimensional data is that the data points are distributed in a union of subspaces (UoS) [10, 9]. If the subspaces are known a priori and represented as a dictionary, then the structure underlying the data can be represented by their coefficients over the dictionary atoms. Specifically, given a dictionary $\mathbf{A}$ of a UoS and a vector $\mathbf{x}$ representing a data sample, the problem becomes to find a few atoms in $\mathbf{A}$ that span the subspace where $\mathbf{x}$ resides in. The problem can be formulated as to identify such atoms by solving the following optimization problem,

$$\begin{aligned} \min_{\mathbf{w}} \quad & \|\mathbf{w}\|_0 \\ \text{s.t.} \quad & \mathbf{A}\mathbf{w} = \mathbf{x}, \end{aligned} \tag{1.1}$$

where the coefficient $\mathbf{w}$ can thus be considered as a compact representation of $\mathbf{x}$ in the UoS. We can relax the ℓ_0-norm with combinatorial nature in (1.1) into ℓ_1-norm so as to have a convex problem.

When such a dictionary $\mathbf{A}$ is unknown, it is still possible to recover the underlying subspace structure when sufficient data samples are present [4, 17, 11]. In particular, we can exploit the property of "self-representation" among the data, that is, each data sample can be represented as a linear combination of other samples from a same subspace. The problem of identifying the self-representation relations can be formulated as a sparse self-representation (SSR) problem as follows,

$$\begin{aligned} \min_{\mathbf{W}} \quad & \|\mathbf{W}\|_1 \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{X}\mathbf{W}, \mathrm{diag}(\mathbf{W}) = 0, \end{aligned} \tag{1.2}$$

where $\mathbf{W}$ should have a block-diagonal structure under certain permutation, and each block corresponds to the data samples from a same subspace. The above problem has also be interpreted as to find clustering structures (i.e., block structures in $\mathbf{W}$) of subspaces and thus also referred to as Sparse Subspace Clustering (SSC) [12].

Sparse modeling methods on high-dimensional data have achieved superior performance on tasks such as clustering and subspace recovery [4, 17, 11]. However, these methods assume linear relations among data and thus cannot effectively model data of nonlinearity. Fortunately, nonlinear data may exhibit linearity when mapped to an implicit higher-dimensional feature space via a kernel function [31], and in the implicit feature space the sparse models can be utilized to recover the linear relations among the data, and thus the nonlinear relations in the original feature space.

In this paper, we propose a kernelized sparse self-representation (KSSR) model that combines sparse models and kernel methods to learn underlying nonlinear structure of a given dataset. We also propose a Kernelized Fast Iterative Soft-Thresholding algorithm, denoted as K-FISTA, to solve the corresponding optimization problem. We demonstrate that the proposed method works well compared to the

*GE global research, {xiao.bian@ge.com}

†Indiana University-Purdue University Indianapolis, {lifen@umail.iu.edu, xning@iupui}

other state-of-the-art methods for nonlinear data clustering purposes. We also apply this method for collaborative filtering [25] in recommender systems, and demonstrate its potential for novel applications.

2 Related Work

The idea of combining sparse models and kernel methods has been explored in previous works [22, 13, 29, 16, 30]. In Patel *et al.* [22], unsupervised learning such as clustering has been conducted via sparse representation in a low-dimensional latent space. The relation between the original data space and the learned latent space is through a projection matrix, which implies that only linear relations among data samples are considered. In Gao *et al.* [13], sparse models with nonlinear kernels integrated have been applied to supervised learning tasks such as image classification. However, an a priori dictionary is required so as to represent data samples in an implicit feature space. Thiagarajan *et al.* [29] has combined sparse models with kernel methods into a dictionary learning problem. Although they can learn a dictionary from the given data, the associated optimization problem is non-convex and computationally very expensive (includes matrix inversion and SVD). Nguyen *et al.* [30] applies sparse dictionary learning on kernelized data, and thus the learned dictionary is non-linear. Similarly to Thiagarajan *et al.* [29], the optimization problem in non-convex and computationally expensive.

Different from the previous work, our method is designed for unsupervised learning tasks. It is able to reveal data relations without a priori information such as a pre-defined dictionary. In addition, it does not involve an unknown dictionary that has to be learned, which introduces non-convexity. Furthermore, we propose an efficient learning algorithm K-FISTA by adapting the state-of-the-art sparse coding algorithm FISTA [3] with kernel functions, which enables us to solve the kernelized sparse self-representation problem efficiently.

3 A Kernelized Sparse Self-Representation Method

Consider a dataset $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, where each column $\mathbf{x}_i \in R^d$ represents a data sample, and assume there exists a kernel function $\Phi : \mathbf{x} \to \mathbf{z}$ such that the UoS structure holds for $\mathbf{Z} = \Phi(\mathbf{X})$, then we learn the linear sparse self-representation relations among $\mathbf{Z}$ by solving the following sparse representation problem,

$$\begin{aligned} \min_{\mathbf{W}} \quad & \|\mathbf{W}\|_1 \\ \text{s.t.} \quad & \Phi(\mathbf{X}) = \Phi(\mathbf{X})\mathbf{W}, \mathrm{diag}(\mathbf{W}) = 0, \end{aligned} \tag{3.3}$$

and thus the nonlinear relations among $\mathbf{X}$. Notice that if Φ is known a priori, then Problem (3.3) remains the same as Problem (1.2).

3.1 The existence of mapping Φ

The feasibility of the formulation in Problem (3.3) depends upon the existence of the mapping Φ. We first prove that a mapping Φ that leads to UoS structures on $\mathbf{Z} = \Phi(\mathbf{X})$ does exist.

The rationale of mapping the input data into a higher-dimensional latent feature space in kernel methods relies on the assumption that holds generally: data points from different classes become linearly separable in a certain higher-dimensional feature space. As a matter of fact, any partition of $d+1$ points in general positions is linearly separable in R^d. Nevertheless, in the setting of Problem (3.3), we need the data clusters in the original feature space fall into different subspaces after mapping. In specific, if manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$ are disjoint in the original feature space, their images after mapping should intersect only at the origin. The following theorem guarantees the existence of such a mapping.

THEOREM 3.1. *Assume $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$ are pairwise-disjoint manifolds embedded in R^d, then there exists a mapping $\Phi : \mathbf{x} \to \mathbf{z}, \forall \mathbf{x} \in R^d$, such that $\mathcal{S}_i \cap \mathcal{S}_j = \{0\}, i \neq j, 1 \leq i, j \leq n$, where $\mathcal{S}_i$ is the minimal subspace satisfying $\Phi(\mathcal{M}_i) \subseteq \mathcal{S}_i$.*

Proof. Given that $\forall i, j, i \neq j, 1 \leq i, j \leq m$, $\mathcal{M}_i \cap \mathcal{M}_j = \emptyset$, we can construct a series of m functions $f_i (1 \leq i \leq m)$ that satisfy

$$f_i(\mathbf{x}) = \begin{cases} 1, & \forall \mathbf{x} \in \mathcal{M}_i \\ 0, & \text{otherwise.} \end{cases}$$

Define a mapping function

$$\Phi(\mathbf{x}) = [f_1(\mathbf{x})\mathbf{x}^\mathsf{T}, f_2(\mathbf{x})\mathbf{x}^\mathsf{T}, \ldots, f_m(\mathbf{x})\mathbf{x}^\mathsf{T}]^\mathsf{T} \in R^{md},$$

then $\forall \mathbf{x} \in \mathcal{M}_i$, we have

$$\Phi(\mathbf{x}) = [0, \ldots, 0, \mathbf{x}^\mathsf{T}, 0, \ldots, 0]^\mathsf{T}, \tag{3.4}$$

where $\mathbf{x}^\mathsf{T}$ is from index $id+1$ to $(i+1)d$ in $\Phi(\mathbf{x})$.

Consider S_i as the minimal subspace containing $\Phi(\mathcal{M}_i), 1 \leq i \leq m$, according to Equation (3.4), we have

$$supp(S_i) \cap supp(S_j) = \emptyset, i \neq j, 1 \leq i, j \leq m, \tag{3.5}$$

which essentially implies that $S_i \cap S_j = \{0\}$.

Note that the condition $\mathcal{S}_i \cap \mathcal{S}_j = \{0\}, i \neq j, 1 \leq i, j \leq m$, is essentially necessary for a self-representation model as Problem (1.2) to achieve perfect decomposition [4]. Theorem 3.1 implies that two data manifolds in the same subspace can be separated after mapping to a higher-dimensional latent feature space. This essentially provides a viable approach to clustering data in the original feature space. When data lacks UoS structures in the original feature space, they may reside in different subspaces in a higher-dimensional latent feature space, and hence can be characterized by a sparse representation model in that space. The feature mapping enables a unified framework of learning Union of Subspaces (UoS) and Union of Manifolds (UoM).

3.2 Kernelized fast iterative soft-thresholding algorithm Identifying an explicit form of the mapping Φ is usually intractable due to the unknown high, even infinite dimensionality of an appropriate feature space. A wise approach to dealing with this issue is to play the "kernel trick" [26, 6]. Specifically, if the solution of Problem (3.3) can be calculated in a way such that only the inner products of data in the higher-dimensional latent feature space are involved, we can work on the latent feature space implicitly via a kernel function. Thus, we propose the following Kernelized Fast Iterative Soft-Thresholding Algorithm (K-FISTA) algorithm to solve Problem 3.3 efficiently.

Since noise is inevitable in practice, we first relax the equality constraint in Problem (3.3), and reformulate the problem into the following form,

$$\begin{aligned} (3.6)\quad \min_{\mathbf{W}} \quad & \lambda\|\mathbf{W}\|_1 + \frac{1}{2}\|\Phi(\mathbf{X})\mathbf{W} - \Phi(\mathbf{X})\|_F^2 \\ \text{s.t.} \quad & \text{diag}(\mathbf{W}) = 0, \end{aligned}$$

where λ balances the fidelity term and the sparsity of $\mathbf{W}$.

Define

$$f(\mathbf{W}) = \frac{1}{2}\|\Phi(\mathbf{X})\mathbf{W} - \Phi(\mathbf{X})\|_F^2.$$

Under the framework of proximal-point methods [7, 5], we solve the following sub-problem in each iteration so as to approximate the objective function in Problem (3.6).

$$\begin{aligned} \mathbf{W}_{k+1} = \arg\min_{\mathbf{W}} \quad & \{f(\mathbf{W}_k) \\ (3.7) \quad & +(\mathbf{W} - \mathbf{W}_k)^\mathsf{T}\nabla f(\mathbf{W}_k) \\ & +\frac{L}{2}\|\mathbf{W} - \mathbf{W}_k\|_F^2 + \lambda\|\mathbf{W}\|_1\}, \end{aligned}$$

where $L > 0$ is a stepsize. In fact, Problem (3.7) has a closed-form solution as follows,

$$(3.8)\qquad \mathbf{W}_{k+1} = \mathcal{T}_{\lambda/L}\left(\mathbf{W}_k - \frac{\nabla f(\mathbf{W}_k)}{L}\right),$$

where $\mathcal{T}_{\lambda/L}(\cdot)$ is the soft-thresholding operator. We expand the gradient of $f(\mathbf{W})$ as

$$(3.9)\qquad \nabla f(\mathbf{W}) = \Phi(\mathbf{X})^\mathsf{T}(\Phi(\mathbf{X})\mathbf{W} - \Phi(\mathbf{X})).$$

Assume $\mathbf{K}_{\mathbf{X},\mathbf{X}} = \Phi(\mathbf{X})^\mathsf{T}\Phi(\mathbf{X})$ is the kernel function, then Equation (3.9) can be represented as

$$\nabla f(\mathbf{W}) = \mathbf{K}_{\mathbf{X},\mathbf{X}}\mathbf{W} - \mathbf{K}_{\mathbf{X},\mathbf{X}},$$

(i.e., kernel tricks) and we hence have Equation (3.8) as

$$(3.10)\qquad \mathbf{W}_{k+1} = \mathcal{T}_{\lambda/L}\left(\mathbf{W}_k - \frac{\mathbf{K}_{\mathbf{X},\mathbf{X}}\mathbf{W} - \mathbf{K}_{\mathbf{X},\mathbf{X}}}{L}\right),$$

that is, $\mathbf{W}$ can be updated using $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ in each iteration. Thus, we can solve Problem (3.6) without knowing the explicit form of $\Phi(\mathbf{X})$.

In order to achieve a good convergence rate in Problem (3.6), we adopt the idea of FISTA algorithm [3] into a kernelized version. FISTA algorithm falls within the category of proximal-point methods and smartly chooses another sequence $\mathbf{Y}_k$ based on information from previous steps instead of using $\mathbf{W}_k$. The resulted kernelized FISTA (K-FISTA) algorithm for solving Problem (3.6) is presented in Algorithm 1.

Algorithm 1 K-FISTA for KSSR

Input: Data matrix $\mathbf{X} \in R^{d\times n}$, $\lambda \in R$
Initialize: $\mathbf{K}_{\mathbf{X},\mathbf{X}}$, $L = 2\|\mathbf{K}_{\mathbf{X},\mathbf{X}}\|_2$, $\mathbf{Y}_1 = \mathbf{W}_0$, $t_1 = 1$, $k = 1$
while not converge **do**
$\mathbf{W}_k = \mathcal{T}_{\lambda/L}\left(\mathbf{Y}_k - \frac{\mathbf{K}_{\mathbf{X},\mathbf{X}}\mathbf{Y}_k - \mathbf{K}_{\mathbf{X},\mathbf{X}}}{L}\right)$
$\text{diag}(\mathbf{W}_k) = \mathbf{0}$
$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
$\mathbf{Y}_{k+1} = \mathbf{W}_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{W}_k - \mathbf{W}_{k-1})$
$k = k + 1$
end while
return $\mathbf{W}_{k-1}$

From a more general perspective, $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ may be interpreted as a similarity metric of $\mathbf{X}$. According to Mercer's theorem [6], $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ is a valid kernel as long as it is positive and semi-definite. Practically, finding an appropriate similarity metric and reformulating it as a kernel is usually more tractable than searching for a suitable nonlinear mapping. We will evaluate the performance K-FISTA using popular kernels in Section 4.

3.2.1 Computational complexity The computational complexity of K-FISTA mainly relates to two parts: 1) the total number of iterations (i.e. the convergence rate) and 2) the computational cost in each iteration. The convergence rate for FISTA algorithm is $O(1/k^2)$ [3], where k is the number of iterations. For each iteration, the computational cost is determined by the soft-thresholding operation, and hence is $O(n^2)$, where n is the number of samples.

The original SSR model as in Problem (1.2) has a computational complexity $O(nd)$, where d is the dimension of the input data space. When $d > n$, that is, the input data space is of high-dimensionality such as for images and time series, K-FISTA is generally more efficient than SSR. In this scenario, K-FISTA with a linear kernel is able to achieve the same solution as that of SSR, but at a lower computational cost.

3.3 Kernelized sparse representation with known dictionaries When a priori dictionary is enforced to model the given data, for example, exemplar samples are given with labels, we therefore has a supervised problem such as classification and regression. We can further extend (3.6) to a sparse model with a known dictionary to solve this problem using a similar efficient algorithm as Algorithm 1. Although the focus of this work is on unsupervised learning, we would like to show that K-FISTA can be easily extended to supervised tasks. Specifically, we can reformulate (3.6) as

$$\min_{\mathbf{W}} \lambda\|\mathbf{W}\|_1 + \frac{1}{2}\|\Phi(\mathbf{D})\mathbf{W} - \Phi(\mathbf{X})\|_F^2, \tag{3.11}$$

where $\mathbf{D}$ is an a-priori dictionary matrix.

The motivation of utilizing a nonlinear mapping Φ in Problem (3.11) is same as that in Problem (3.6). In particular, we represent each data point using atoms of the dictionary $\mathbf{D}$ in the higher-dimensional feature space, where $\mathbf{D}$ and $\mathbf{X}$ may exhibit linear relations.

Similarly, define $g(\mathbf{W}) = \frac{1}{2}\|\Phi(\mathbf{D})\mathbf{W} - \Phi(\mathbf{X})\|_F^2$, and it follows that

$$\begin{aligned}\nabla g(\mathbf{W}) &= \Phi(\mathbf{D})^\mathsf{T}(\Phi(\mathbf{D})\mathbf{W} - \Phi(\mathbf{X})) \\ &= \mathbf{K}_{\mathbf{D},\mathbf{D}}\mathbf{W} - \mathbf{K}_{\mathbf{D},\mathbf{X}}.\end{aligned}$$

We therefore can solve $\mathbf{W}$ iteratively using the kernel function $\mathbf{K}_{\mathbf{D},\mathbf{D}}$ and $\mathbf{K}_{\mathbf{D},\mathbf{X}}$ similarly as in Equation (3.10). The details are presented in Algorithm 2.

Algorithm 2 K-FISTA with a known dictionary

Input: Data matrix $\mathbf{X} \in R^{d\times n}$, Dictionary $\mathbf{D} \in R^{d\times m}$, $\lambda \in R$

Initialize: $\mathbf{K}_{\mathbf{D},\mathbf{D}}$, $\mathbf{K}_{\mathbf{D},\mathbf{X}}$, $L = 2\|\mathbf{K}_{\mathbf{D},\mathbf{D}}\|_2$, $\mathbf{Y}_1 = \mathbf{W}_0, t_1 = 1, k = 1$

while not converge **do**

$$\mathbf{W}_k = \mathcal{T}_{\lambda/L}\left(\mathbf{Y}_k - \frac{\mathbf{K}_{\mathbf{D},\mathbf{D}}\mathbf{Y}_k - \mathbf{K}_{\mathbf{D},\mathbf{X}}}{L}\right)$$

$$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$$

$$\mathbf{Y}_{k+1} = \mathbf{W}_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{W}_k - \mathbf{W}_{k-1})$$

$k = k + 1$

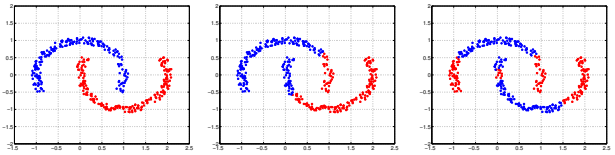
end while

return $\mathbf{W}_{k-1}$

4 Experimental Results

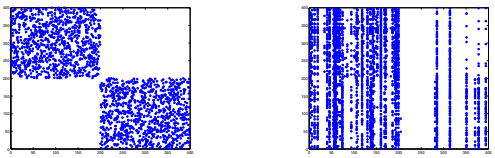
In this section, we present the experimental results of KSSR on two different tasks: clustering and recommendation. For clustering, we conduct experiments on both synthetic data and real-world data to evaluate the performance of KSSR. In particular, we focus on testing whether the sparse coefficient matrix $\mathbf{W}$ reflects the structure of the given data correctly. For recommendation, we conduct item-based collaborative filtering using $\mathbf{W}$ and evaluate whether accurate recommendations can be produced.

4.1 Clustering results on synthetic data We first evaluate KSSR on synthetic nonlinear datasets and compare it with spectral clustering (SC) [20] and SSR for clustering [11]. The synthetic dataset I is generated with 400 data samples distributed in two disconnected $2/3$ circles with uniformly distributed errors as in Fig. 1a. Each arch is considered as a cluster. The synthetic dataset II is generated with 600 data samples distributed in two Archimedean spirals with uniformly distributed errors as in Fig. 2a. Each spiral is considered as a cluster. The two synthetic datasets are nonlinear.



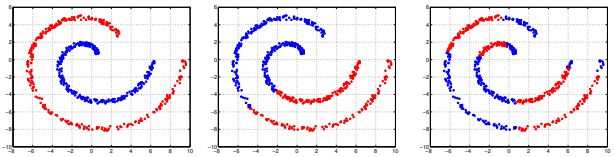
(a) KSSR (b) SC (c) SSR

Figure 1: Clustering results on synthetic dataset I



(a) KSSR (b) SSR

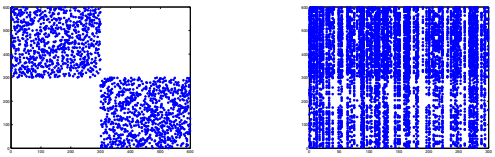
Figure 2: $\mathbf{W}$ of KSSR and SSR from synthetic dataset I



(a) KSSR (b) SC (c) SSR

Figure 3: Clustering results on synthetic dataset II

In Fig. 1 and Fig. 3, we present the clustering results of the three methods KSSR, SC and SSR on the two synthetic datasets, respectively, where different predicted clusters are indicated by different colors. We use a same RBF kernel for both KSSR and SC. In Fig. 1, when the two arches are close to each other, SC begins to fail. SSR can hardly differentiate the two arches. However, KSSR has a more robust performance on this dataset and well separates the two arches. Similar performance can be observed on the



(a) KSSR (b) SSR

Figure 4: $\mathbf{W}$ of KSSR and SSR from synthetic dataset II

synthetic dataset II as in Fig. 3. The results demonstrate that under the highly nonlinear setting, SC and SSR both fail to provide a good cut of the disjoint data manifolds. However, KSSR is able to address the nonlinearity and thereby separate the two data manifolds apart as demonstrated in Fig. 1a and Fig. 3a, respectively. We can also observe the clear clustering structures from the sparse coefficients $\mathbf{W}$ from KSSR. As shown in Fig. 2a and Fig. 4a, $\mathbf{W}$ learned from KSSR exhibits distinct block-diagonal structures for both datasets, where each block corresponds to the data points in one cluster. $\mathbf{W}$ from SSR fails to capture the underlying data structures as in Fig. 2b and Fig. 4b due to the lack of linearity in these two datasets. The results demonstrate that, although highly nonlinear in the original data space, after implicitly mapping to a latent feature space, the data points indeed reside in different subspaces and exhibit linear relations within each cluster.

SSR is typically not suitable for data in a low-dimensional space, due to the lack of UoS structures. The above results demonstrate that our method can analyze low-dimensional data as long as the data distribution satisfies Theorem 3.1. In these two examples, data points from both clusters reside in a two-dimensional space. However, the RBF kernel can implicitly map the two clusters into different subspaces, and therefore facilitate the clustering of nonlinear data.

4.2 Clustering results on real data We test the KSSR model on both low-dimensional real-world datasets and high-dimensional real-world datasets. The datasets used and their properties are presented in Table 1.

Table 1: Real datasets used for clustering

dataset	dimension	class	size
iris	4	3	150
wine	13	3	178
ORL face	10304	40	400

4.2.1 Clustering on UCI iris and UCI wine We compare KSSR and other methods on two datasets from UCI machine learning repository: the UCI iris dataset and the UCI wine dataset [2]. We use clustering accuracy and the normalized mutual information (NMI) [1] to evaluate the performance of each method. In both metrics, a higher value indicates a better clustering performance. For these two low-dimensional datasets, we compare KSSR to Kmeans [27], SC and SSR.

For the iris dataset, there are three types of iris plants with a total of 150 data points, and each data point is a 4-dimensional vector. The performance of KSSR compared to other clustering methods is presented in Table 2, where Kmeans and SC use the known number of clusters as their input parameter, and SSR and KSSR use the optimal parameters that are determined via grid search. Table 2 shows that KSSR has a better clustering performance under both evaluation metrics.

Table 2: Clustering performance on UCI iris dataset

metric	Kmeans	SC	SSR	KSSR
accuracy (%)	89.33	89.33	82.67	**90.67**
NMI (%)	75.82	74.96	65.72	**80.57**

Bold numbers correspond to the best performance.

The wine dataset is composed of chemical features of three different types of wines. Each data point is a 13-dimensional vector. We present the performance of KSSR compared to other clustering methods in Table 3 and the results show that KSSR achieves the highest accuracy and NMI compared to other three methods.

Table 3: Clustering performance on UCI wine dataset

metric	Kmeans	SC	SSR	KSSR
Accuracy (%)	96.07	96.07	81.46	**99.44**
NMI (%)	85.03	85.03	50.72	**97.33**

Bold numbers correspond to the best performance.

4.2.2 Clustering on face images We evaluate the performance of KSSR on the high-dimensional ORL database of faces [24]. This database includes face images from 40 subjects with 10 images of each. Images are taken under different facial expressions and various lighting conditions. Each image is of the dimension $112 \times 92 = 10304$. In this experiment, we compare KSSR with Kmeans, SSR and nonnegative matrix factorization (NMF) [15] as NMF is among the most effective methods to cluster images. Similarly, Kmeans and SC use the known number of clusters as their input parameter, and SSR and KSSR select optimal parameters via grid search.

We present the clustering results under different numbers of subjects in Table 4 and Table 5, where each subject is considered as a cluster of the images. For each number of subjects (clusters) in Table 4 and Table 5, we repeat the experiments 20 times by randomly choosing subjects from the entire dataset, and then calculate the average performance of

Table 4: Clustering performance (accuracy (%)) on ORL face

# of subjects	Kmeans	NMF	SSR	KSSR
2	94.25	83.75	**97.50**	96.75
3	85.17	72.33	91.33	**92.67**
4	75.50	65.63	87.38	**95.38**
5	77.60	66.10	78.90	**89.00**
6	78.67	63.42	80.08	**90.83**
7	72.21	62.50	81.29	**89.86**
8	73.56	63.19	78.00	**89.56**
9	75.50	58.78	73.50	**86.28**
10	74.85	58.25	77.05	**87.65**

Bold numbers correspond to the best performance.

Table 5: Clustering performance (NMI (%)) on ORL face

# of subjects	Kmeans	NMF	SSR	KSSR
2	85.89	58.27	**92.36**	91.87
3	82.46	61.01	85.91	**88.33**
4	82.67	59.60	86.02	**92.64**
5	84.37	66.09	81.55	**87.84**
6	85.30	65.66	82.89	**90.01**
7	83.36	69.08	83.97	**90.64**
8	86.10	71.87	84.02	**91.37**
9	85.80	69.23	82.43	**89.29**
10	85.70	70.67	84.64	**90.07**

Bold numbers correspond to the best performance.

different methods as shown in these two tables.

Under both metrics, KSSR has the best average performance. In fact, it has the highest scores in all cases except the 2-subject case. Compared to the concept factorization in [18] using the same dataset, we achieve a substantial improvement on the performance from less than 80% to around 90% under both metrics. The setting in this paper is exactly same as [18] except that the performance is the average of 20-time experiments instead of 10.

In this set of experiments, SSR has good performance primarily due to the high-dimensionality of the data space. Specifically, images of one subject approximately span a low-dimensional subspace. However, nonlinear transformations, for example, expression variations, beards and glasses, make the images deviate from the underlying subspaces, and therefore the distribution of images of each subject has some nonlinear components. KSSR can address the nonlinearity and therefore outperforms SSR model in almost all cases.

4.3 Recommendation results on real data The learned coefficient matrix $\mathbf{W}$ from Problem (3.3) can be considered as a measurement of item-item "similarities", though it is not strictly a similarity metrics. That is, if one data point can be used to represent another data point (from a same manifold), then they share certain "similarities" represented by $\mathbf{W}$. With respect to the original data space, such "similarities" can be considered as from a certain latent "similarity" function. Under this interpretation, we can use the coefficient matrix $\mathbf{W}$ for item-based collaborative filtering (CF) in recommender systems (RS). The basic idea of item-based CF is that, for each user in the system, we find the items that are similar to the items that have been purchased by the user, and aggregate their similarities to generate recommendations for the user [8]. CF has been a very effective method in recommender systems, and significant efforts on CF are dedicated to generating informative item-item similarities [23].

Table 6: Real datasets used for recommendation

datasets	#users	#items	#nnz	rsize	csize	density
ML100K	943	1,682	100,000	106.0	59.5	6.30%
Yelp	13,574	6,896	89,608	6.6	13.0	0.10%

#nnz represents the number of purchases of users on items; rsize, csize and density represent the average row density, the average column density and the average matrix density, respectively.

In specific, we conduct *top-n* recommendation, that is, for each user, we generate a ranked list of n items for recommendation. The real datasets used in this set of experiments are presented in Table 6. The ML100K dataset corresponds to movie ratings and was obtained from the MovieLens research project [1]. The Yelp dataset is a subset of the academic version of Yelp user-business rating and review dataset downloaded from Yelp [2]. The users who reviewed at least 3 businesses and the corresponding businesses were selected to construct the Yelp dataset.

We apply 5-time Leave-One-Out cross validation (LOOCV) to evaluate the recommendation performance. In each run, each dataset is split into a training set and a testing set by randomly selecting one of the purchased items of each user into the testing set. We follow the item-based CF protocol as in Deshpande *et al.* [8] to generate the recommendations. We compare the recommendation list of each user and the item of that user in the testing set and measure the performance using Hit Rate (HR) and the Average Reciprocal Hit-Rank (ARHR) [8] defined as follows,

$$\text{HR} = \frac{\#\text{hits}}{\#\text{users}}, \tag{4.12}$$

where #users is the total number of users, and #hits is the number of users whose item in the testing set is recommended (i.e., hit); and

$$\text{ARHR} = \frac{1}{\#\text{users}} \sum_{i=1}^{\#\text{hits}} \frac{1}{p_i}, \tag{4.13}$$

where if a user has a hit, p is the position of the recommended item in the recommendation list. ARHR measures how the

[1] http://grouplens.org/datasets/movielens/

[2] http://www.yelp.com/academic_dataset

hits are ranked in the recommendation list. Higher HR and ARHR values indicate better recommendation performance.

Table 7 presents the performance of KSSR on ML100K and Yelp. KSSR shows smooth performance change over the parameter space, which is a favorable property for CF.

Table 7: Recommendation performance of KSSR

ML100K				Yelp			
σ	λ	HR	ARHR	σ	λ	HR	ARHR
50	0.0100	0.324	0.144	28	0.40	0.258	0.114
50	0.0010	0.336	0.156	28	0.30	0.314	0.133
50	0.0001	0.337	0.155	28	0.20	0.105	0.044
45	0.0100	0.327	0.146	24	0.30	0.278	0.119
45	0.0010	**0.341**	**0.157**	24	0.20	**0.324**	**0.132**
45	0.0001	0.340	0.155	24	0.10	0.100	0.041

σ is the parameter for RBF. Bold numbers correspond to the best performance.

Table 8 represents the performance of KSSR-based CF compared with the state-of-the-art *top-n* recommendation methods including item-based k-NN method itemkNN [8], weighted regularized matrix factorization WRMF [14], and a sparse linear method SLIM [21]. In this set of experiments, we recommend n=10 items.

Table 8: Recommendation performance comparison

dataset	metric	itemkNN	WRMF	SLIM	KSSR
ML100K	HR	0.287	0.327	0.343	0.341
	ARHR	0.124	0.133	0.147	0.157
Yelp	HR	0.257	0.327	0.348	0.324
	ARHR	0.092	0.131	0.147	0.132

Table 8 demonstrates that the performance of KSSR is very comparable with those from the state-of-the-art *top-n* recommendation methods WRMF and SLIM. In addition, KSSR can be superior to these methods when not only the recommendation performance is the concern, but also the understanding of the involved items is critical, as KSSR discovers some nonlinear structures that cannot be revealed from the state-of-the-art *top-n* recommendation methods.

5 Conclusion

We propose in this paper a novel sparse model KSSR to recover the underlying nonlinear structures of a dataset. By integrating the kernel function into the sparse representation model, we first map nonlinear data into an implicit feature space, and then utilize the self-representation property of the data in the feature space to partition the data into different manifolds. Our method achieves superior performance on clustering problems on multiple datasets, and shows good adaptability on both low-dimensional data and high-dimensional data. Our method also demonstrate strong potential for its novel application in collaborative filtering in recommender systems. Future work include systematical ways to find suitable kernel functions for sparse representation, more understanding on the behaviors of KSSR for CF and its further improvement, etc. Additionally, it would be interesting to further modify the model to adapt potential outliers and data corruptions.

References

[1] LNF Ana and Anil K Jain. Robust data clustering. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–128. IEEE, 2003.

[2] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[3] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[4] Xiao Bian and Hamid Krim. Robust Subspace Recovery via Bi-Sparsity Pursuit. *ArXiv e-prints*, March 2014.

[5] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[6] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[7] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11):1413–1457, 2004.

[8] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22:143–177, January 2004.

[9] Michael Elad. Sparse and redundant representation modeling: What next? *IEEE Signal Processing Letters*, 19:922–928, December 2012.

[10] Yonina C Eldar and Gitta Kutyniok. *Compressed sensing: theory and applications*. Cambridge University Press, 2012.

[11] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797. IEEE, 2009.

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

[13] Shenghua Gao, Ivor Wai-Hung Tsang, and Liang-Tien Chia. Sparse representation with kernels. *Image Processing, IEEE Transactions on*, 22(2):423–434, 2013.

[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[15] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[16] Bao-Di Liu, Yu-Xiong Wang, Bin Shen, Yu-Jin Zhang, and Martial Hebert. Self-explanatory sparse representation for image classification. In *Computer Vision–ECCV 2014*, pages 600–616. Springer, 2014.

[17] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013.

[18] Haifeng Liu, Zheng Yang, and Zhaohui Wu. Locality-constrained concept factorization. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1378, 2011.

[19] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696. ACM, 2009.

[20] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[21] Xia Ning and George Karypis. SLIM: Sparse linear models for top-N recommender systems. In *IEEE International Conference on Data Mining*, ICDM'11, 2011.

[22] Vishal M Patel, Hien Van Nguyen, and René Vidal. Latent space sparse subspace clustering. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 225–232. IEEE, 2013.

[23] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[24] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 138–142. IEEE, 1994.

[25] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[26] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[27] George AF Seber. *Multivariate observations*, volume 252. John Wiley & Sons, 2009.

[28] Mahdi Soltanolkotabi and Emmanuel J Candes. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, 40(4):2195–2238, 2012.

[29] Jayaraman J. Thiagarajan, Karthikeyan Natesan Ramamurthy, and Andreas Spanias. Multiple Kernel Sparse Representations for Supervised and Unsupervised Learning. *IEEE Transactions on Image Processing*, 23:2905–2915, July 2014.

[30] Hien Van Nguyen, Vishal M Patel, Nasser M Nasrabadi, and Rama Chellappa. Design of non-linear kernel dictionaries for object recognition. *Image Processing, IEEE Transactions on*, 22(12):5123–5135, 2013.

[31] Vladimir Naumovich Vapnik. *Statistical learning theory*, volume 2. Wiley New York, 1998.