KERNEL SPARSE SUBSPACE CLUSTERING

Vishal M. Patel[*]

Center for Automation Research
UMIACS
University of Maryland, College Park, MD

René Vidal[†]

Center for Imaging Science
Department of Biomedical Engineering
Johns Hopkins University, Baltimore, MD

ABSTRACT

Subspace clustering refers to the problem of grouping data points that lie in a union of low-dimensional subspaces. One successful approach for solving this problem is sparse subspace clustering, which is based on a sparse representation of the data. In this paper, we extend SSC to non-linear manifolds by using the kernel trick. We show that the alternating direction method of multipliers can be used to efficiently find kernel sparse representations. Various experiments on synthetic as well real datasets show that non-linear mappings lead to sparse representation that give better clustering results than state-of-the-art methods.

***Index Terms*—** Subspace clustering, sparse subspace clustering, kernel methods, non-linear subspace clustering.

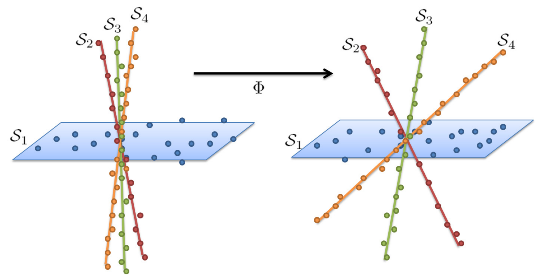

Fig. 1: Overview of the proposed non-linear subspace clustering methods. Using the kernel trick, data is transformed onto a high-dimensional feature space so that better sparse representations can be found for subspace clustering.

1. INTRODUCTION

Many practical computer vision and image processing applications require processing and representation of high-dimensional data. Often these high-dimensional data can be represented by a low-dimensional subspace. For instance, it is well known that the set of face images under all possible illumination conditions can be well approximated by a 9-dimensional linear subspace [1]. Similarly, trajectories of a rigidly moving object in a video [2] and hand written digits with different variations [3] also lie in low-dimensional subspaces. One can view the collection of data from different classes as samples from a union of low-dimensional subspaces. In subspace clustering, given the data from a union of subspaces, the objective is to find the number of subspaces, their dimensions, the segmentation of the data and a basis for each subspace [4].

Various algorithms have been proposed in the literature for subspace clustering. Some of these algorithms are iterative in nature [5], [6] while the others are based on spectral clustering [7], [8], [9], [10]. Statistical [11] and algebraic [12], [13] approaches have also been proposed in the literature for subspace clustering. In particular, sparse representation and low-rank approximation-based methods for subspace clustering [14], [15], [10], [16], [17], [18], [19], [20], [21] have gained a lot of traction in recent years. These methods find a sparse or low-rank representation of the data and build a similarity graph on the sparse or low-rank coefficient matrix for segmenting the data. One of the advantages of these methods is that they are robust to noise and occlusion. Furthermore, some of these approaches do not require the knowledge of the dimensions and the number of subspaces. In particular, the Sparse Subspace Clustering (SSC) algorithm [10], [16] and Low-Rank Representation (LRR) [14] based algorithms are well supported by theoretical analysis [22] [17], [18] and provide state-of-the-art results on many publicly available datasets such as the Hopkins155 benchmark motion segmentation dataset [23].

While the linear model is a good approximation, in practice many datasets are better modeled by non-linear manifolds. In this case, algorithms such as SSC are no longer applicable. One approach to dealing with nonlinear manifolds is to solve for a weighted sparse representation, as proposed in [24]. However, while this approach leads to better embeddings, its doesn't improve the clustering performance. Another approach to dealing with non-linear manifolds is to use kernel methods. For instance, kernel-based sparse representations have been exploited before in the context of compressed sensing [25] , sparse coding [26] and dictionary learning [27], [28]. It has been shown that the non-linear mapping using the kernel trick can group the data with the same distribution and make them linearly separable. As a result, sparse representation of the data can be found easily and the representation error can be reduced significantly.

Motivated by the success of non-linear representations in various computer vision and image understanding applications, we propose a non-linear extension of SSC method for manifold clustering. It is shown that by using the kernel trick, one can obtain a sparse representation of the data in a high-dimensional feature space that leads to better clustering results. Fig. 1 presents an overview of our method. The proposed kernel SSC (KSSC) method is based on the classical alternating direction method of multipliers.

This paper is organized as follows. A brief background of the SSC method is provided in Section 2. Details of our nonlinear extension of SSC method is covered in Section 3. Experimental results are presented in Section 4 and Section 5 concludes the paper with a brief summary and discussion.

[*]The work of VMP was partially supported by an Office of Naval Research grant N00014-12-1-0124.

[†]The work of RV was partially supported by National Science Foundation grant 11-1218709.

2. SPARSE SUBSPACE CLUSTERING

In this section, we provide a brief background on sparse subspace clustering. Let $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N] \in \mathbb{R}^{D\times N}$ be a collection of N signals $\{\mathbf{y}_i \in \mathbb{R}^D\}_{i=1}^N$ drawn from a union of n linear subspaces $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_n$ of dimensions $\{d_\ell\}_{\ell=1}^n$ in $\mathbb{R}^D$. Let $\mathbf{Y}_\ell \in \mathbb{R}^{D\times N_\ell}$ be a sub-matrix of $\mathbf{Y}$ of rank d_ℓ with $N_\ell > d_\ell$ points that lie in $\mathcal{S}_\ell$ with $N_1 + N_2 + \cdots + N_n = N$. Given $\mathbf{Y}$, the task of subspace clustering is to cluster the signals according to their subspaces.

It is easy to see that each data point in $\mathbf{Y}$ can be efficiently represented by a linear combination of at most d_ℓ other points in $\mathbf{Y}$. That is, one can represent $\mathbf{y}_i$ as follows

$$\mathbf{y}_i = \mathbf{Y}\mathbf{c}_i, \ \ c_{ii} = 0, \ \ \|\mathbf{c}_i\|_0 \le d_\ell$$

where $\mathbf{c}_i = [c_{i1}, c_{i2}, \cdots, c_{iN}]^T \in \mathbb{R}^N$ are the coefficients and $\|\mathbf{x}\|_0$ is the sparsity measure that counts the number of non-zero elements in $\mathbf{x}$. Often $N_\ell \gg d_\ell$. As a result the following ℓ_1-minimization problem is solved to obtain the coefficients

$$\min \|\mathbf{c}\|_1 \text{ such that } \mathbf{y}_i = \mathbf{Y}\mathbf{c}_i, \ c_{ii} = 0, \tag{1}$$

where $\|x\|_1 = \sum_{i=1}^N |x_i|$ is the ℓ_1-norm of $\mathbf{x} \in \mathbb{R}^N$. Considering all the data points $i = 1, \cdots, N$, in matrix form, the above optimization problem can be rewritten as

$$\min \|\mathbf{C}\|_1 \ \text{ subject to } \ \mathbf{Y} = \mathbf{Y}\mathbf{C}, \ \ \text{diag}(\mathbf{C}) = \mathbf{0}, \tag{2}$$

where $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_N] \in \mathbb{R}^{N\times N}$ is the coefficient matrix whose column $\mathbf{c}_i$ is the sparse representation vector corresponding to $\mathbf{y}_i$, $\text{diag}(\mathbf{C}) \in \mathbb{R}^N$ is the vector containing the diagonal elements of $\mathbf{C}$ and $\mathbf{0} \in \mathbb{R}^N$ is an N-dimensional vector containing zeros as its elements.

In the case where the data is contaminated by some arbitrary noise $\mathbf{Z}$, i.e., $\mathbf{Y} = \mathbf{Y}\mathbf{C} + \mathbf{Z}$, the following problem is solved to obtain $\mathbf{C}$

$$\min \|\mathbf{C}\|_1 + \lambda_1 \|\mathbf{Y} - \mathbf{Y}\mathbf{C}\|_F^2 \quad \text{s. t.} \quad \text{diag}(\mathbf{C}) = \mathbf{0}, \tag{3}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. In practice, the data may lie in a union of affine subspaces. In this case, the following problem can be solved to obtain the sparse coefficients

$$\begin{aligned} &\min \|\mathbf{C}\|_1 + \lambda_1 \|\mathbf{Y} - \mathbf{Y}\mathbf{C}\|_F^2, \\ &\qquad\qquad \text{s. t. } \ \text{diag}(\mathbf{C}) = \mathbf{0}, \ \ \mathbf{C}^T\mathbf{1} = \mathbf{1}, \end{aligned} \tag{4}$$

where $\mathbf{1}$ is a vector of dimension N containing ones as its elements.

The above problems can be efficiently solved by using the classical alternating direction method of multipliers (ADMM) [16]. In SSC, once $\mathbf{C}$ is found, spectral clustering methods [29] are applied on the affinity matrix $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^T$ to obtain the segmentation of the data $\mathbf{Y}$ into $\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_n$.

3. NON-LINEAR EXTENSION OF SSC

Let $\Phi : \mathbb{R}^D \to \mathcal{H}$ be a mapping from the input space to the reproducing kernel Hilbert space $\mathcal{H}$. Let $\mathcal{K}_{\mathbf{YY}} \in \mathbb{R}^{N\times N}$ be a positive semidefinite kernel Gram matrix whose elements are computed as

$$[\mathcal{K}_{\mathbf{YY}}]_{i,j} = [\langle \Phi(\mathbf{Y}), \Phi(\mathbf{Y})\rangle_{\mathcal{H}}]_{i,j} \quad = \Phi(\mathbf{y}_i)^T \Phi(\mathbf{y}_j) = \kappa(\mathbf{y}_i, \mathbf{y}_j),$$

where $\kappa : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is the kernel function and

$$\Phi(\mathbf{Y}) = [\Phi(\mathbf{y}_1), \Phi(\mathbf{y}_2), \cdots, \Phi(\mathbf{y}_N)].$$

Some commonly used kernels include polynomial kernels $\kappa(\mathbf{x}, \mathbf{y}) = \langle(\mathbf{x}, \mathbf{y}\rangle + a)^b$ and Gaussian kernels $\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\sigma\|\mathbf{x} - \mathbf{y}\|^2\right)$, where a, b and σ are the parameters of the kernel functions. Equipped with the above notations, in what follows, we provide non-linear extension of SSC method.

3.1. Kernel SSC

Non-linear version of the optimization problem (4) can be written as follows

$$\begin{aligned} &\min \|\mathbf{C}\|_1 + \lambda_1 \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{C}\|_F^2, \\ &\qquad\qquad \text{s. t. } \ \text{diag}(\mathbf{C}) = \mathbf{0}, \ \ \mathbf{C}^T\mathbf{1} = \mathbf{1}. \end{aligned} \tag{5}$$

In terms of the kernel matrix $\mathcal{K}_{\mathbf{YY}}$, this can be equivalently written as

$$\begin{aligned} &\min \|\mathbf{C}\|_1 + \lambda_1 Tr(\mathcal{K}_{\mathbf{YY}} - 2\mathcal{K}_{\mathbf{YY}}\mathbf{C} + \mathbf{C}^T\mathcal{K}_{\mathbf{YY}}\mathbf{C}), \\ &\qquad\qquad \text{s. t. } \ \text{diag}(\mathbf{C}) = \mathbf{0}, \ \ \mathbf{C}^T\mathbf{1} = \mathbf{1}. \end{aligned} \tag{6}$$

Note that the above formulation explicitly depends on the kernel matrix $\mathcal{K}_{\mathbf{YY}}$ but not on the mapping Φ. This problem can be efficiently solved using the ADMM method.

3.1.1. ADMM method for solving (5)

Consider the following optimization problem

$$\begin{aligned} &\min_{\mathbf{A},\mathbf{C}} \|\mathbf{C}\|_1 + \lambda_1 \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\|_F^2, \\ &\qquad\qquad \text{s. t. } \ \mathbf{A} = \mathbf{C} - \text{diag}(\mathbf{C}), \ \ \mathbf{A}^T\mathbf{1} = \mathbf{1}, \end{aligned} \tag{7}$$

where we have introduced an auxiliary matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. The solution to this optimization problem coincides with the solution of (5). By adding the two penalty terms corresponding to the two constrains in (7), we get the following optimization problem

$$\begin{aligned} &\min_{\mathbf{A},\mathbf{C}} \|\mathbf{C}\|_1 + \lambda_1 \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\|_F^2 \\ &\qquad + \frac{\rho}{2}\|\mathbf{A} - \mathbf{C} + \text{diag}(\mathbf{C})\|_F^2 + \frac{\rho}{2}\|\mathbf{A}^T\mathbf{1} - \mathbf{1}\|_2^2 \\ &\qquad \text{s. t. } \ \mathbf{A} = \mathbf{C} - \text{diag}(\mathbf{C}), \ \ \mathbf{A}^T\mathbf{1} = \mathbf{1}. \end{aligned} \tag{8}$$

Both (7) and (8) have the same solutions. We can write the Lagrangian formulation of (8) by introducing a vector $\boldsymbol{\delta} \in \mathbb{R}^N$ and a matrix $\triangle \in \mathbb{R}^{N\times N}$ as follows

$$\begin{aligned} \mathcal{L}(\mathbf{C}, \mathbf{A}, \boldsymbol{\delta}, \triangle) &= \min_{\mathbf{A},\mathbf{C}} \|\mathbf{C}\|_1 + \lambda_1 \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\|_F^2 \\ &+ \frac{\rho}{2}\|\mathbf{A} - \mathbf{C} + \text{diag}(\mathbf{C})\|_F^2 + \frac{\rho}{2}\|\mathbf{A}^T\mathbf{1} - \mathbf{1}\|_2^2 \\ &+ Tr(\triangle^T(\mathbf{A} - \mathbf{C} + \text{diag}(\mathbf{C}))) + \boldsymbol{\delta}^T(\mathbf{A}^T\mathbf{1} - \mathbf{1}). \end{aligned} \tag{9}$$

In the ADMM method, variables are optimized one at a time while keeping the other variables fixed. In what follows, we describe each of the suboptimization problems in detail.

Update step for $\mathbf{A}$*:* With fixed $\mathbf{C}_k, \boldsymbol{\delta}_k, \triangle_k$, $\mathbf{A}_{k+1}$ is obtained by minimizing $\mathcal{L}$ with respect to $\mathbf{A}$. As a result, $\mathbf{A}_{k+1}$ is obtained by solving the following linear system of equations

$$\begin{aligned} &(\lambda_1\mathcal{K}_{\mathbf{YY}} + \rho\mathbf{I} + \rho\mathbf{1}\mathbf{1}^T)\mathbf{A}_{k+1} \\ &\qquad = \lambda_1\mathcal{K}_{\mathbf{YY}} + \rho(\mathbf{1}\mathbf{1}^T + \mathbf{C}_k - \text{diag}(\mathbf{C}_k)) - \mathbf{1}\boldsymbol{\delta}_k^T - \triangle_k. \end{aligned}$$

Note that the update on $\mathbf{A}$ depends on the kernel matrix $\mathcal{K}_{\mathbf{YY}}$.

Update step for $\mathbf{C}$: To find $\mathbf{C}_{k+1}$, we fix $\mathbf{A}_k, \boldsymbol{\delta}_k, \triangle_k$ and minimize $\mathcal{L}$ with respect to $\mathbf{C}$. This gives us the following solution

$$\mathbf{C}_{k+1} = \mathbf{J} - \text{diag}(\mathbf{J}), \;\; \mathbf{J} = \mathcal{T}_{\frac{1}{\rho}}\left(\mathbf{A}_{k+1} + \frac{\triangle_k}{\rho}\right).$$

Here, $\mathcal{T}_\alpha(.)$ denotes the shrinkage thresholding operator acting on each element of the given matrix defined as $\mathcal{T}_\alpha(x) = (|x| - \alpha)_+ \text{sgn}(x)$, where the operator $(.)_+$ returns its argument if it is non-negative and returns zero otherwise.

Update steps for $\boldsymbol{\delta}$, $\triangle$: Having fixed $\mathbf{C}_k, \mathbf{A}_k$, gradient ascent with step size of ρ is performed to update $\boldsymbol{\delta}$, $\triangle$ as

$$\begin{aligned}\boldsymbol{\delta}_{k+1} &= \boldsymbol{\delta}_k + \rho(\mathbf{A}_{k+1}^T\mathbf{1} - \mathbf{1}) \\ \triangle_{k+1} &= \triangle_k + \rho\left(\mathbf{A}_{k+1} - \mathbf{C}_{k+1} + \text{diag}(\mathbf{C}_{k+1})\right).\end{aligned}$$

These steps are repeated until, $\|\mathbf{A}_k^T\mathbf{1} - \mathbf{1}\|_\infty \leq \epsilon$, $\|\mathbf{A}_k - \mathbf{C}_k\|_\infty \leq \epsilon$, $\|\mathbf{A}_k - \mathbf{A}_{k-1}\|_\infty \leq \epsilon$. Different steps for the ADMM optimization of (5) are summarized in Algorithm 1.

Algorithm 1: ADMM algorithm for solving (5)

```
Input: $\mathcal{K}_{\mathbf{YY}}, \lambda_1, \rho$.
Initialization:
- Set MaxIter=$10^4$ and Terminate ← False.
- Set $\mathbf{C}_0 = \mathbf{0}, \mathbf{A}_0 = \mathbf{0}, \boldsymbol{\delta}_0 = \mathbf{0}$ and $\triangle_0 = \mathbf{0}$.
while (Terminate == False) do
- Update $\mathbf{A}_{k+1}$ by solving the following system of equations:
  $(\lambda_1\mathcal{K}_{\mathbf{YY}} + \rho\mathbf{I} + \rho\mathbf{1}\mathbf{1}^T)\mathbf{A}_{k+1}$
  $= \lambda_1\mathcal{K}_{\mathbf{YY}} + \rho(\mathbf{1}\mathbf{1}^T + \mathbf{C}_k - \text{diag}(\mathbf{C}_k)) - \mathbf{1}\boldsymbol{\delta}_k^T - \triangle_k$
- Update $\mathbf{C}_{k+1}$ as $\mathbf{C}_{k+1} = \mathbf{J} - \text{diag}(\mathbf{J})$, where
  $\mathbf{J} = \mathcal{T}_{\frac{1}{\rho}}\left(\mathbf{A}_{k+1} + \frac{\triangle_k}{\rho}\right)$
- Update $\boldsymbol{\delta}_{k+1}$ as $\boldsymbol{\delta}_{k+1} = \boldsymbol{\delta}_k + \rho(\mathbf{A}_{k+1}^T\mathbf{1} - \mathbf{1})$
- Update $\triangle_{k+1}$ as
$\triangle_{k+1} = \triangle_k + \rho\left(\mathbf{A}_{k+1} - \mathbf{C}_{k+1} + \text{diag}(\mathbf{C}_{k+1})\right)$
- $k \leftarrow k + 1$
  if $\|\mathbf{A}_k^T\mathbf{1} - \mathbf{1}\|_\infty \leq \epsilon$ and $\|\mathbf{A}_k - \mathbf{C}_k\|_\infty \leq \epsilon$ and
     $\|\mathbf{A}_k - \mathbf{A}_{k-1}\|_\infty \leq \epsilon$ or $k \geq$ maxIter
then
Terminate ← True
  end if
end while
Output: $\hat{\mathbf{C}} = \mathbf{C}_k$.
```

Similar to the linear SSC method, once the sparse coefficient matrix $\mathbf{C}$ is found in the high dimensional feature space, spectral clustering is applied on the affinity matrix $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^T$ to obtain the segmentation of the data.

4. EXPERIMENTAL RESULTS

In this section, we evaluate our proposed method on both synthetic and real datasets. We compare our method with several state-of-the-art subspace clustering algorithms such as SSC [16], Low-Rank Representation (LRR) [14], Low-Rank Subspace Clustering (LRSC) [15], Local Subspace Affinity (LSA) [9] and Spectral Curvature Clustering (SCC) [7]. The selection of parameters is done through a 5-fold cross-validation. All the experiments are done on an OS X system with 2.6 GHz Intel Core i7 processor using Matlab. Subspace clustering error is used to measure the performance of different algorithms. It is defined as

$$\text{subspace clustering error} = \frac{\#\text{of misclassified points}}{\text{total}\#\text{of points}} \times 100.$$

4.1. Simulated Data

In this section, we present results of our KSSC method on synthetic data. We will show that by mapping the data onto a feature space one can obtain better sparse codes for SSC. The smallest principle angle, θ_{ij}, between two subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$ is defined as

$$\cos(\theta_{i,j}) = \max_{\mathbf{a}_i \in \mathcal{S}_i, \mathbf{a}_j \in \mathcal{S}_j} \frac{\mathbf{a}_i^T\mathbf{a}_j}{\|\mathbf{a}_i\|_2\|\mathbf{a}_j\|_2}.$$

It was shown in [16] that when the smallest principle angle between subspaces and the number of data points in each subspace is small, the SSC clustering error increases. We study the performance of our method when the data in each subspace and the smallest principle angle between subspaces are small.

We follow the same experimental setting as in [16]. We consider $n = 3$ subspaces of dimension $d = 4$ embedded in $D = 55$ dimensional space. We generate the subspace bases $\{\mathbf{T}_i \in \mathbb{R}^{D\times d}\}_{i=1}^3$ such that rank $([\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3]) = 2d$. Also, the subspaces are generated such that $\theta_{12} = \theta_{23} = \theta$. Furthermore, we generate the same number of points, N_g, in each subspace at random and change the value of N_g.

For a fixed value of d, we change the minimum angle between subspaces, θ, as well as the number of points in each subspace N_g. For each pair of (θ, N_g), we compute the subspace clustering error. Since the performance of SSC and KSSC methods are based on how well the sparse coefficients are found, we also calculate the subspace sparse recovery error. For the data points $\{\mathbf{y}_i\}_{i=1}^{3N_g}$, the sparse recovery error E_{SR} is given by

$$E_{SR} = \frac{1}{3N_g}\sum_{i=1}^{3N_g}\left(1 - \frac{\|\mathbf{c}_{iq_i}\|_1}{\|\mathbf{c}_i\|_1}\right),$$

where $\mathbf{c}_i^T = [\mathbf{c}_{i1}{}^T, \mathbf{c}_{i2}{}^T, \mathbf{c}_{i3}{}^T]$ represents the sparse coefficients of $\mathbf{y}_i \in \mathcal{S}_{q_i}$ and $\mathbf{c}_{ij}$ corresponds to the points in $\mathcal{S}_j$.

We vary the smallest principle angle between subspaces and the number of points in each subspace as $\theta \in [6, 60]$ degrees and $N_g \in [d+1, 30d]$, respectively. For each pair (θ, N_g), we calculate the average subspace clustering error as well as the average E_{SR} over 20 trials. In each trial we randomly generate data points and subspaces. Polynomial kernel with parameters $a = 0$ and $b = 2$ is used in this experiment. Results of this experiment are shown in Fig. 2.

When θ and N_g decrease, both the sparse recovery and clustering errors of SSC and KSSC methods increase. Also, the clustering error is highly dependent on the sparse recovery error and both errors follow the same pattern. In other words, clustering results are highly dependent on how well the sparse coefficients are recovered. By comparing the decay of errors, one can see that in the case where both θ and N_g are small, the KSSC method performs better than the SSC method. The error decays faster in the case of KSSC than SSC. This can be explained by the fact that in KSSC polynomial kernel is able to map the data in the high-dimensional feature space in a way that the principle angle between the subspaces increases.

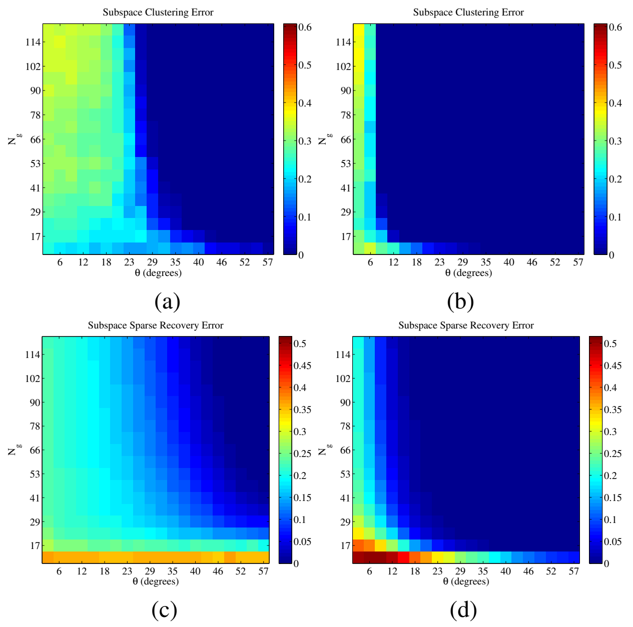

Fig. 2: The average subspace clustering and subspace-sparse recovery errors of the SSC method are shown in (a) and (c), respectively. The average subspace clustering and subspace-sparse recovery errors of the KSSC method are shown in (b) and (d), respectively.

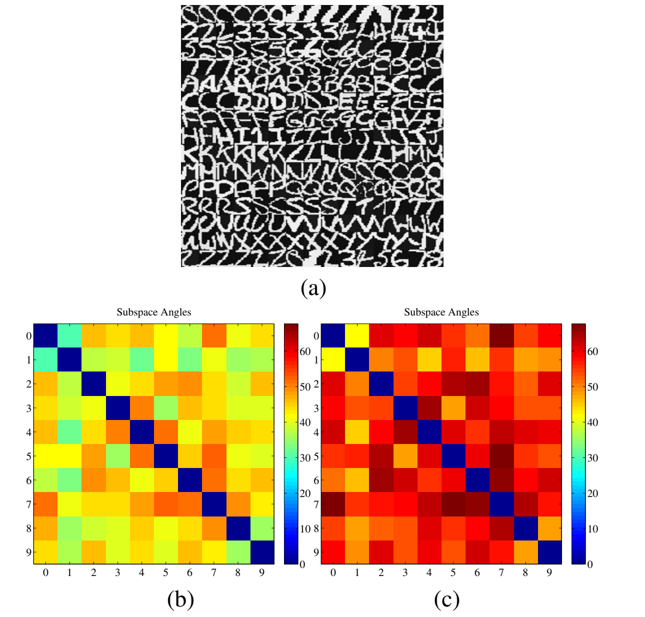

Fig. 3: (a) Samples digits from the Binary Alphadigits dataset. (b) Original subspace angles. (c) Subspace angles after the data has been transformed into a feature space. Note that the non-linear mapping generally increases the distance between two subspaces .

4.2. Handwritten Digits Clustering

In this section, we present results of different subspace clustering methods on the Binary Alphadigits dataset .[1] The Binary Alphadigits dataset contains binary digits of 0 through 9 and capital A through Z. Each digit is of size 20×16. There are 39 examples of each class. Samples digits from this dataset are shown in Fig. 3(a). This dataset is challenging because of the intraclass variations among different digits. For instance, the digits 0, 2 and 5 look very similar to the letters O, Z and S, respectively.

Similar to the experimental set up in [16], to study the effect of the number of digits in the clustering performance of different methods, we divide the 36 digits into 4 groups, where the first three groups correspond to digits 1-10, A-J, K-T, and the fourth group corresponds to digits U-Z. For each of the first three groups, we consider all choices of $n \in \{2, 3, 5, 8, 10\}$ digits and for the last group we consider all choices of $n \in \{2, 3, 5\}$. We apply various subspace clustering algorithms on each trial and record their results. For KSSC, we used Gaussian and polynomial kernels with different parameters.

The results of different subspace clustering methods are shown in Table 1. In Table 1, $P(a, b)$ denotes a polynomial kernel with parameters a, b and $G(\sigma)$ denotes a Gaussian kernel with parameter σ. As can be seen from this table that KSSC provides comparable results to the other linear subspace clustering methods. In particular, KSSC provides the best results on $n = 2, 3$ and 5, when a polynomial kernel with parameters $a = 3, b = 2$ is used for clustering. In Fig. 3(b) and (c), we show subspace angles corresponding to the numerical digits 0-9 in the original and in the feature space. As can be seen from this figure, non-linear mapping does increase the angle between two subspaces which in turn helps in improving the overall clustering error.

[1] Available at http://www.cs.toronto.edu/∼roweis/data.html

Algorithms (2 digits)	LRSC	LSA	SSC	LRR	KSSC P(3,2)	KSSC P(2,0.2)	KSSC G(10)	KSSC G(7)
Mean	15.81	10.70	5.70	7.76	5.42	5.40	6.13	5.93
Median	8.97	3.85	2.56	3.84	2.56	2.56	2.56	2.56
Algorithms (3 digits)	LRSC	LSA	SSC	LRR	KSSC P(3,2)	KSSC P(2,0.2)	KSSC G(10)	KSSC G(7)
Mean	25.65	22.69	13.58	14.21	12.85	13.30	14.54	13.64
Median	25.64	22.22	8.54	11.11	7.69	8.54	9.40	8.55
Algorithms (5 digits)	LRSC	LSA	SSC	LRR	KSSC P(3,2)	KSSC P(2,0.2)	KSSC G(10)	KSSC G(7)
Mean	37.77	33.81	23.26	23.34	22.64	23.00	24.50	23.84
Median	37.95	33.85	25.12	23.59	23.08	23.85	27.18	26.15
Algorithms (8 digits)	LRSC	LSA	SSC	LRR	KSSC P(3,2)	KSSC P(2,0.2)	KSSC G(10)	KSSC G(7)
Mean	47.98	40.76	30.00	30.50	31.06	31.24	31.38	31.19
Median	48.08	40.06	30.01	30.44	32.05	32.05	31.73	31.09
Algorithms (10 digits)	LRSC	LSA	SSC	LRR	KSSC P(3,2)	KSSC P(2,0.2)	KSSC G(10)	KSSC G(7)
Mean	50.77	42.65	32.14	33.67	33.85	35.30	33.16	32.48
Median	51.03	41.28	32.82	32.56	34.36	36.15	34.10	33.33

Table 1: Clustering errors on the Alphadigits dataset.

5. CONCLUSION

We have presented a non-linear subspace clustering algorithm that exploits sparsity of data in high dimensional feature space by extending the SSC subspace clustering algorithm through an appropriate choice of kernel. Experimental results indicate that exploiting non-linear parse representation in a high-dimensional feature space can provide better clustering than the traditional subspace clustering approaches.

6. REFERENCES

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

[2] J. P. Costeira and T. Kanade, "A multibody factorization method for independently moving objects," *International Journal of Computer Vision*, vol. 29, no. 3, pp. 159–179, 1998.

[3] Trevor Hastie and Patrice Y. Simard, "Metrics and models for handwritten character recognition," *Statistical Science*, vol. 13, no. 1, pp. 54–65.

[4] R. Vidal, "Subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.

[5] J. Ho, M. H. Yang, J. Lim, K. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[6] T. Zhang, A. Szlam, and G. Lerman, "Median k-flats for hybrid linear modeling with many outliers," in *Workshop on Subspace Methods*, 2009.

[7] G. Chen and G. Lerman, "Spectral curvature clustering (SCC)," *International Journal of Computer Vision*, vol. 81, no. 3, pp. 317–330, 2009.

[8] A. Goh and R. Vidal, "Segmenting motions of different types by unsupervised manifold clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[9] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *European Conf. on Computer Vision*, 2006, p. 94106.

[10] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2790–2797.

[11] M. A. Fischler and R. C. Bolles, "Ransac random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 26, pp. 381–395, 1981.

[12] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1–15, 2005.

[13] K. Kanatani, "Motion segmentation by subspace separation and model selection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2001, vol. 2, pp. 586–591.

[14] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *International Conference on Machine Learning*, 2010.

[15] P. Favaro, R. Vidal, and A. Ravichandran, "A closed form solution to robust subspace estimation and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[16] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2013.

[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*, vol. 35, no. 1, pp. 171–184, 2013.

[18] R. Vidal and P. Favaro, "Low rank subspace clustering (LRSC)," *Pattern Recognition Letters*, 2013.

[19] Qiang Qiu and Guillermo Sapiro, "Learning transformations for clustering and classification," *Preprint*, 2013.

[20] Guangcan Liu and Shuicheng Yan, "Latent low-rank representation for subspace segmentation and feature extraction," in *IEEE International Conference on Computer Vision*, 2011, pp. 1615–1622.

[21] V. M. Patel, H. V. Nguyen, and R. Vidal, "Latent space sparse subspace clustering," in *International Conference on Computer Vision*, 2013.

[22] M. Soltanolkotabi and E. J. Candes, "A geometric analysis of subspace clustering with outliers," *Annals of Statistics*, vol. 40, no. 4, pp. 2195–2238, 2011.

[23] R. Tron, R. Vidal, and A. Ravichandran, "A benchmark for the comparison of 3-d motion segmentation algorithms," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[24] E. Elhamifar and R. Vidal, "Sparse manifold clustering and embedding," in *Neural Information Processing Systems*, 2011.

[25] Hanchao Qi and Shannon Hughes, "Using the kernel trick in compressive sensing: Accurate signal recovery from fewer measurements," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, may 2011, pp. 3940 –3943.

[26] L. Zhang, W-D. Zhou, P-C. Chang, J. Liu, Z. Yan, T. Wang, and F-Z. Li, "Kernel sparse representation-based classifier," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1684 –1695, April 2012.

[27] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Kernel dictionary learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012, pp. 2021–2024.

[28] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Design of non-linear kernel dictionaries for object recognition," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5123–5135, 2013.

[29] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Neural Information Processing Systems*, 2002, vol. 2, pp. 849–856.