Transformed Subspace Clustering

Jyoti Maggu, Angshul Majumdar and Emilie Chouzenoux

Abstract— Subspace clustering assumes that the data is separable into separate subspaces. Such a simple assumption, does not always hold. We assume that, even if the raw data is not separable into subspaces, one can learn a representation (transform coefficients) such that the learnt representation is separable into subspaces. To achieve the intended goal, we embed subspace clustering techniques (locally linear manifold clustering, sparse subspace clustering and low rank representation) into transform learning. The entire formulation is jointly learnt; giving rise to a new class of methods called transformed subspace clustering (TSC). In order to account for non-linearity, kernelized extensions of TSC are also proposed. To test the performance of the proposed techniques, benchmarking is performed on image clustering and document clustering datasets. Comparison with state-of-the-art clustering techniques shows that our formulation improves upon them.

Index Terms—Transform Learning, Subspace Clustering, Image Clustering, Document Clustering.

1 Introduction

THE problem of clustering is well known. It studies how signals are naturally grouped together. Perhaps the simplest and most widely used clustering technique is the K-means [1]. It groups the samples such that the total distance of the data points within the cluster are minimized. The problem is NP hard, and hence is usually solved greedily.

One of the limitations of K-means is that it operates on the raw data and hence fails to capture non-linear relationships. The simple fix to that is the kernel K-means [2]. The concept remains the same as in any kernel trick; operationally instead of Euclidean distances between the samples, its kernelized version is used for K-means.

Related to the kernel K-means is spectral clustering [2, 3]. The kernelized data matrix is the same as the affinity matrix for spectral clustering. It just generalizes the kernels to be any similarity measure and not necessarily based on Mercer kernels.

Subspace clustering techniques [4] assume that the samples from the same cluster will lie in the same subspace. Operationally, it involves expressing each data point as a linear combination of other data points. These linear weights serve as inputs for creating the affinity matrix. The factorization based techniques prevalent in clustering (such as [5-7]) technically belong to the subspace clustering paradigm.

In the past, it has been found that instead of applying subspace clustering on the raw data, a projection can be learnt such that the clustering is carried out in the projected domain. In [8, 9] a tight-frame was learnt from the data along with the subspace clustering formulation. The fundamental assumption behind such formulations is that even if the original data do not fall on separate sub-spaces, their projected versions will.

Our work is based on similar assumptions. Instead of applying subspace clustering on the original space, we will learn the subspace clustering in the transformed space. Therefore, even if the data cannot be segmented / clustered in the original domain, its transformed representation can be clustered into separate subspaces. The work proposes to incorporate three variants of subspace clustering – i) Locally linear manifold clustering (LLMC), ii) sparse subspace clustering (SSC), and iii) low rank representation (LRR).

We compared our proposed formulations with state-of-the-art representation learning and clustering techniques. We show that our method improves over the rest by a considerable margin in terms the clustering metrics used here.

2 Literature Review

2.1 Subspace Clustering

Subspace clustering techniques like locally linear manifold clustering (LLMC) [11, 12], sparse subspace clustering (SSC) [13-15] and low rank representation (LRR) [16, 17] express the samples as a linear combination of other samples. This is expressed as,

$$x_i = X_{i^c} c_i, \forall i \text{ in } \{1,\ldots,n\} \tag{1}$$

Here x_i ($\in \mathbb{R}$) denotes the i^{th} sample and X_{i^c} ($\in \mathbb{R}$) all other samples; c_i ($\in \mathbb{R}$) is the corresponding linear weight vector.

For all the sub-space clustering techniques the general learning formulation can be expressed as follows,

$$\min_{c_i} \left\| x_i - X_{i^c} c_i \right\|_2^2 + R(c_i), \forall i \text{ in } \{1,\ldots,n\} \tag{2}$$

Here R is the regularization term. Depending on its nature there are three formulations. For LLMC there is no regularization. For sparse subspace clustering, R is a sparsity promoting l_1-norm [13] or l_0-norm [14, 15]. For LRR, R is a low-rank penalty usually in the form of nuclear norm.

For each formulation, once the coefficient matrix $C = [\breve{c}_1, \ldots, \breve{c}_n]$ is obtained for all n samples, the affinity matrix is computed. Note that $\breve{c}_i$ ($\in \mathbb{R}$) is defined from c_i (

- *J. Maggu and A. Majumdar are with Indraprastha Institute of Information-Technology, New Delhi, India 110020. E-mail: {jyotim and angshul}@iiitd.ac.in.*
- *E. Chouzenoux is with University of Paris-Est, LIGM, UMR CNRS 8019, France and CVN, CentraleSupélec, INRIA Saclay, France. E-mail: emilie.chouzenoux@univ-mlv.fr.*

$\in \mathbb{R}$) by putting zero in the ith position. There is no unique definition to the affinity matrix; the only requirement is that it needs to be symmetric. Several variants have been proposed [4]. For example one option can be –

$$A = |C| + |C^T| \tag{3}$$

This is usually used in SSC.

Another option for LRR is to form the affinity matrix from the scaled left singular values of C; this is defined as,

$$A_{ij} = \left(\left[\tilde{\iota}\ \tilde{}\ \right]_{ij}\right) \tag{4}$$

where $\tilde{\iota}$ and $C = USV^T$.

Yet another way to generate the affinity matrix (usually for LLMC) is by.

$$A = C + C^T - C^T C \tag{5}$$

Once the affinity matrix is defined (by using any suitable formula), one needs to segment the clusters. Usually spectral clustering algorithm (Normalized-Cuts) [18] is used for this purpose.

Related to subspace clustering are the matrix factorization based approaches. Some of them have been referred before. For a review one can peruse [19]. The basic idea there in, is to identify arbitrary subspaces from the data by factoring it.

2.2 Transform Learning

Transform learning analyses the data by learning a transform / basis to produce coefficients. Mathematically this is expressed as,

$$TX = Z \tag{6}$$

Here T is the transform, X is the data and Z the corresponding coefficients. The following transform learning formulation was proposed in [4]:

$$\min_{T,Z} \|TX - Z\|_F^2 + \lambda\left(\|T\|_F^2 - \log\det T\right) + \mu\|Z\|_1 \tag{7}$$

The parameters (λ and μ) are positive. The factor $-\log\det T$ imposes a full rank on the learned transform; this prevents the degenerate solution ($T=0, Z=0$). The additional penalty $\|T\|_F^2$ is to balance scale.

In [4], an alternating minimization approach was proposed to solve the transform learning problem. This is given by:

$$Z \leftarrow \min_{Z} \|TX - Z\|_F^2 + \mu\|Z\|_1 \tag{8a}$$

$$T \leftarrow \min_{T} \|TX - Z\|_F^2 + \lambda\left(\|T\|_F^2 - \log\det T\right) \tag{8b}$$

Updating the coefficients (8a) is straightforward using one step of soft thresholding,

$$Z \leftarrow signum(TX) \cdot \max\left(0, abs(TX) - \mu\right) \tag{9}$$

Here '$\cdot$' indicates element-wise product.

The update for the transform (8b) also has a closed form solution. This is given as,

$$XX^T + \lambda I = LL^T \tag{10a}$$

$$L^{-1}XZ^T = USV^T \tag{10b}$$

$$T = 0.5U\left(S + (S^2 + 2\lambda I)^{1/2}\right)V^T L^{-1} \tag{10c}$$

The proof for convergence of such an alternating update algorithm can be found in [21].

3 Proposed Formulations

We have discussed in the previous sections the existing concepts of subspace clustering and transform learning. Our contribution in this paper is to embed three subspace clustering formulations, associated to three distinct choices for the regularization term R, into the transformed space, that is, instead of learning the affinity matrix from the raw data, we propose to learn it from the (transform) coefficient space.

The trend of learning representations for machine learning has been used rampantly in supervised problems. For example consider [22, 23]. In [22] it is assumed that even if the data cannot be discriminated in the original domain, they can be learnt to be discriminative in the feature domain using dictionary learning. Similarly, in [23], it is assumed that even though one cannot linearly project the raw samples to their corresponding class labels, the learnt dictionary coefficients can be. Similar ideas are echoed in [24] using the transform learning approach. As mentioned before, similar ideas have been explored for clustering as well [18, 19]. In [25], it has been assumed that even if the data is not clustered on subspaces, its deeply (stacked autoencoder) learnt representation will be. Our work follows the same trend; we assume that even if the original data do not lie on separate subspaces, the (transform) learnt coefficients will lie on different subspaces and hence the ensuing affinity matrix can be segmented for clustering.

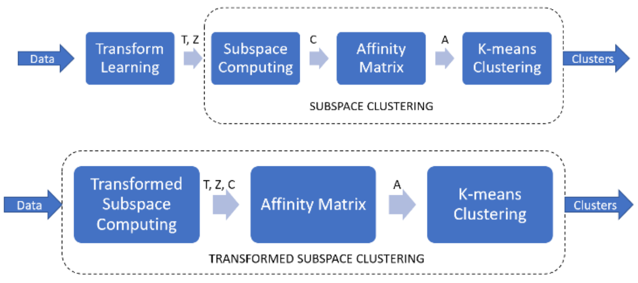

Fig. 1. Top – Piecemeal Solution. Bottom – Proposed Solution

A naïve piecemeal solution (Fig. 1, top) would be to learn the transform on the data and then use the coefficients as inputs for subspace clustering. But such a piecemeal formulation will not yield the best results. This can be seen from [25] and [26]. In [26], a deep representation is first learnt and then a third-party clustering algorithm is used on the learnt representation; the results improve in [25] when the deep representation and the clustering are jointly learnt.

We propose to formulate a joint solution instead (Fig. 1, bottom). Mathematically, our formulation is expressed as,

$$\min_{T,Z,C} \|TX - Z\|_F^2 + \lambda\left(\|T\|_F^2 - \log\det T\right) + \mu\|Z\|_1 + \gamma\sum_i \|z_i - Z_{i^c}c_i\|_2^2 + R(C) \tag{11}$$

Alternating minimization [27] approach is used for solving (11). It can be segregated into the following sub-problems.

$$\text{P1:} \min_{T} \|TX - Z\|_F^2 + \lambda\left(\|T\|_F^2 - \log\det T\right)$$

$$\text{P2:}\min_{Z} \|TX-Z\|_F^2 + \mu\|Z\|_1 + \gamma\sum_i \|z_i - Z_{i^c}c_i\|_2^2$$

$$\text{P3:}\min_{C} \sum_i \|z_i - Z_{i^c}c_i\|_2^2 + R(C)$$

The update for the transform (P1) remains the same as in (10). The update for the transform coefficients can also be shown to be the same as in (9). Indeed we have,

$$\min_{Z} \|TX-Z\|_F^2 + \mu\|Z\|_1 + \gamma\|Z(I-C)\|_F^2$$

$$\Rightarrow \min_{Z} \|X^TT^T - Z^T\|_F^2 + \mu\|Z^T\|_1 + \gamma\left\|(I-C)^T Z^T\right\|_F^2 \quad (12)$$

$$\Rightarrow \min_{Z} \left\| \begin{pmatrix} X^TT^T \\ 0 \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\gamma}(I-C)^T \end{pmatrix} Z^T \right\|_F^2 + \mu\|Z^T\|_1$$

The update for P3 will be dependent on the regularization term *R(C)*. For transformed locally linear manifold clustering (TLLMC) there is no regularization, hence it takes the form:

$$\min_{C} \sum_i \|z_i - Z_{i^c}c_i\|_2^2 \quad (13)$$

This can be solved for each c_i via a pseudo-inverse operation.

For transformed sparse subspace clustering (TSSC), the regularization is a sparsity enhancing penalty. Here, we make use of the l_1-norm, so that P3 becomes equivalent to

$$\min_{C} \|Z - ZC\| + \mu\|C\|_1 \quad (14)$$

This is a standard l_1-minimization for which we have used the iterative soft thresholding algorithm [28].

For transformed low rank representation (TLLR) formulation, a low rank penalty is imposed on *C*. Here we will consider the nuclear norm penalty (defined as the sum of singular values of the matrix), so that P3 reads

$$\min_{C} \|Z - ZC\| + \mu\|C\|_* \quad (15)$$

The above problem can be solved using singular value shrinkage [29].

Once the matrix C is obtained, the affinity matrix can be created using either of (3), (4) or (5); here we have retained (3) since it appears to yield the best practical results. Spectral clustering is then applied to the affinity matrix in order to segment the data *X*.

It is worth noting that, assuming each sub-problem is solved in an exact manner, the proposed alternating optimization method is guaranteed to converge to a stationary point of our objective function [27].

3.1 Kernelization

An efficient strategy to handle non-linearity in data in machine learning is to employ the so-called kernel trick. If the (transform) learnt coefficients are not linearly separable into subspaces, we assume that projecting them non-linearly to a higher dimension can make them separable into subspaces. This can be achieved by kernelizing the transform subspace clustering formulation. Kernelizing the basic transform learning has been proposed in [30]. In this work, we incorporate subspace clustering into the kernel transform learning formulation.

In kernel transform learning, a non-linear version of the data is represented in terms of a transform made up of linear combination of non-linear version of itself. This is expressed as:

$$\underbrace{\text{[illegible]}}_{\text{[illegible]}}\text{[illegible]}) = Z \quad (17)$$

One can immediately identify the kernel defined by $K = \varphi(X)^T\varphi(X)$. This allows us to express the kernel version of transform learning in the known (transform learning) form:

$$BK = Z \quad (18)$$

Here, the kernel matrix *K* plays the role of the data matrix *X* in the original transform learning formulation; *B* is the linear weights (similar to transform) that needs to be learnt and *Z* is the corresponding coefficient matrix.

Under this notation, we can formulate all our versions of transformed subspace clustering using the generic notation:

$$\begin{aligned} &\min_{B,Z,C} \|BK-Z\|_F^2 + \lambda\left(\|B\|_F^2 - \log\det B\right) \\ &+\mu\|Z\|_1 + \gamma\sum_i \|z_i - Z_{i^c}c_i\|_2^2 + R(C) \end{aligned} \quad (19)$$

The solution for (19) remains the same as in the previous linear case. We can expect improvements over the basic transform learning version with the kernel trick, since the later accounts for non-linearities.

4 Experimental Evaluation

4.1 Image Clustering

In this sub-section we compare our method with two state-of-the-art techniques in image clustering. The first one is a deep learning based approach called deep sparse subspace clustering (DSC) [25]. The second one is the sparse subspace clustering is based on orthogonal matching pursuit (OMP) [14, 15]. We have also compared our proposed (jointly learnt) solution with the piecemeal solution (discussed in section 3).

We follow the experimental protocol from [25, 15]. Experiments were carried out on the COIL20 (object recognition) [31] and Extended YaleB (face recognition) [32] datasets. The COIL20 database contains 1,440 samples distributed over 20 objects. The used YaleB consists of 2,414 samples from 38 individuals. For both the datasets DSIFT (dense scale invariant feature transform) features were extracted. They were further reduced by PCA to a dimensionality of 300. We experiment on these input features; on top of that we also conduct experiments on raw pixel values. Since the ground truth (class labels) for these datasets are available, clustering accuracy was measured in terms of Accuracy, NMI (normalized mutual information), ARI (adjusted rand index), Precision and F-score. The results are shown in Table 1 (COIL20) and Table 2 (YaleB). Since the last stage of clustering involves K-means, we ran the experiment 100 times and report the average values.

The parametric settings for the methods compared against have been taken from the respective papers. For our proposed technique, we have kept *λ=μ=0.1* and *γ=1*. TLLMC does not require specification of any other parameter. TSC has *μ=0.1* as the sparsity promoting term and

TLLR has *μ=0.01* as the rank deficiency term. The algorithms are robust to these parametric values.

TABLE 1
Comparison showing improvement of proposed variants over the state-of-the-art on COIL 20

Metric	DSIFT								RAW							
	DSC	OMP	TLLMC		TSSC		TLRR		DSC	OMP	TLLMC		TSSC		TLRR	
			Jt.	PM	Jt.	PM	Jt.	PM			Jt.	PM	Jt.	PM	Jt.	PM
Accuracy	.85	.82	**.90**	.88	**.90**	.88	.79	.75	.76	.92	**.98**	.95	**.97**	.92	.78	.72
NMI	.91	.50	**.97**	.86	**.98**	.86	.89	.80	.81	.76	**.94**	.90	**.90**	.86	.83	.74
ARI	.84	.50	**.86**	.82	**.88**	.83	.78	.71	.68	.72	**.94**	.91	**.92**	.87	.72	.67
Precision	.82	.85	**.85**	.78	**.88**	.79	.69	.67	.68	.96	**.99**	.92	**.97**	.80	.72	.66
F-measure	.85	.81	**.90**	.84	**.92**	.86	.79	.72	.70	.90	**.97**	.90	**.96**	.87	.74	.70

*Jt. – Jointly Learnt; PM - piecemeal

TABLE 2
Comparison showing improvement of proposed variants over the state-of-the-art on Yale B

Metric	DSIFT								RAW							
	DSC	OMP	TLLMC		TSSC		TLRR		DSC	OMP	TLLMC		TSSC		TLRR	
			Jt.	PM	Jt.	PM	Jt.	PM			Jt.	PM	Jt.	PM	Jt.	PM
Accuracy	.88	.98	**.98**	.94	**.98**	.94	.81	.79	.74	.94	**.97**	.95	**.99**	.95	.69	.66
NMI	.90	.97	**.97**	.95	**.98**	.94	.89	.85	.87	.80	**.85**	.82	**.94**	.90	.74	.72
ARI	.83	.95	**.96**	.90	**.96**	.91	.73	.71	69	.82	**.91**	.89	**.96**	.91	.75	.74
Precision	.79	.97	**.97**	.91	**.98**	.91	.65	.65	.65	.92	**.95**	.91	**.98**	.92	.72	.70
F-measure	.83	.94	**.92**	.88	**.95**	.92	.74	.71	.69	.87	**.90**	.88	**.95**	.92	.74	.71

*Jt. – Jointly Learnt; PM - piecemeal

TABLE 3
Comparison of Different Kernels on proposed variants for COIL 20

Metric	DSIFT					Raw				
	Linear	Poly-2	Poly-3	Laplacian	Gaussian	Linear	Poly-2	Poly-3	Laplacian	Gaussian
Accuracy	.90	.81	.80	.90	**.93**	**.98**	.82	.72	.95	.95
NMI	.97	.91	.89	.95	**.97**	**.94**	.87	.81	.93	.92
ARI	.86	.81	.78	.86	**.87**	**.94**	.81	.80	.92	.91
Precision	.85	.80	.77	.84	**.85**	**.99**	.89	.76	.93	.91
F-measure	.90	.83	.80	.89	**.90**	**.97**	.83	.82	.95	.94

TABLE 4
Comparison of Different Kernels on proposed variants for YALE B

	DSIFT					Raw				
	Linear	Poly-2	Poly-3	Laplacian	Gaussian	Linear	Poly-2	Poly-3	Laplacian	Gaussian
Accuracy	.98	.92	.91	.98	**.98**	**.97**	.88	.76	.96	.96
NMI	.97	.87	.95	.96	**.97**	**.85**	.83	.80	.82	.81
ARI	.96	.95	.88	.96	**.96**	**.91**	.90	.85	.89	.88
Precision	.97	.90	.87	.97	**.98**	**.95**	.87	.79	.95	.95
F-measure	.92	.86	.85	.92	**.92**	**.90**	.81	.76	.90	.90

From Tables 1 and 2, we also observe that the TLLR formulation does not yield good results. This follows from the findings in [25]; there in it was find that LRR based formulations yield poor results on these datasets. This maybe because well performing TLLR formulations accounts for outliers, which the basic version does not. Our formulation is based on the basic version and consequently yields poor results.

Our joint TSSC and TLLMC formulations always yields the best results (except for DSIFT features where OMP based SSC sometimes yields slightly better results than TLLMC).

The aforesaid results are shown for linear kernels. We have discussed in III.A how our method can be kernelized. In the following Tables 3 and 4, we show results of kernelized versions for COIL 20 and Yale B respectively. Note that these results are shown for the jointly learnt variants since they yield better results than their corresponding piecemeal versions.

The results are shown for some typical kernels like linear, Gaussian, Laplacian and polynomial kernels of order 2 and 3. We show results on the TLLMC algorithm since this is the basic subspace clustering algorithm without any prior (sparsity or rank deficiency). The results show that for DSIFT features, the RBF kernel yields the best results, but for the raw pixel values, the linear one still is the best. Results from polynomial kernels are worse than the simple linear technique; the results deteriorate with the increase in the order of the polynomial. The Laplacian kernel is slightly worse than the Gaussian one on average.

All the experiments were run on an Intel i7 processor with 32 GB RAM running a 64 bit Windows 10. The proposed techniques and OMP was based on Matlab, DSC was based on Python. The run-times are shown in Table 5.

TABLE 5
Comparison of Runtime in Seconds

Technique	Coil 20	Yale B
DSC	62.1	60.6
OMP	4.7	3.9
TLLMC	11.0	8.6
TLLMC-Kernel	44.3	37.9
TSSC	12.1	10.7
TSSC-Kernel	50.0	41.6
TLRR	15.6	13.2
TLRR-Kernel	57.9	48.2

The OMP based sparse subspace clustering is understandably the fastest one. Our transformed versions (with linear kernel) are relatively slower because of the additional computational cost of updating the transform in every iteration. Among the linear techniques, TLLMC is the fastest as it has no regularization term. TSC is slightly slower owing to the requirement of the thresholding step in every iteration. TLRR is even slower, because one needs to threshold the singular values; and computing singular values in every iteration is time consuming. The kernelized versions are slower, because the size of the kernel matrix is larger than the original data matrix (larger number of samples than dimensions). DSC is the slowest of them all, since it is a deep technique requiring updating of multiple layers of autoencoders.

4.2 Document Clustering

Our second example focuses on document clustering. For this application, we follow the protocol defined in (the most recent) [33]. For our experiments, we use three data sets TDT2 corpus [34], Reuters-21578 corpus [34], and 20 Newsgroup [34].

The TDT2 English document data set includes six months of material drawn on a daily basis from six English language news sources. In this set, the total number of samples is 9394, the feature dimension is 36771, and the number of clusters is 30. The Reuters-21578 document set is a collection of manually categorized newswire stories from Reuters Ltd. In this set, the total number of samples is 8293, the feature dimension is 18933, and the number of clusters is 65. The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents. In this set, the total number of samples is 18846, the feature dimension is 26214, and the number of clusters is 20.

Following [33], we report the result in terms of two metrics namely entropy and purity. For good clustering, one requires small entropy and high purity. In [33], the metrics are reported by varying the number of clusters from 2 to 10. We follow the same protocol.

Our proposed method has been compared with [33] and with [35]; to the best of our knowledge these are two of the most recent works in this area and are known to yield the best possible results on document clustering.

We found that the kernelized versions yield better results than the linear counterparts; and the best results are obtained from the Gaussian kernel. These results are shown here.

TABLE 6
Comparison showing improvement of proposed method over existing techniques on TDT2

Clusters	Entropy (lower is better)					Purity (higher is better)				
	CFAN [33]	SNMF-PCA [35]	TLLMC	TSSC	TLLR	CFAN [33]	SNMF-PCA [35]	TLLMC	TSSC	TLLR
2	.0000	.0000	.0000	**.0000**	.0000	1.0000	1.0000	1.0000	**1.0000**	1.0000
4	.0000	.0159	.0000	**.0000**	.1659	1.0000	.9956	1.0000	**1.0000**	.9229
6	.0526	.0889	.0063	**.0013**	.0431	.9435	.8954	.9481	**.9963**	.8917
8	.0552	.0941	.0329	**.0465**	.0506	.9476	.8801	.9490	**.9013**	.9661
10	.0808	.0685	.0188	**.0178**	.0631	.9153	.9224	.9265	**.9775**	.9254
Avg.	.0312	.0582	.0126	**.0103**	.0682	.9682	.9285	.9673	**.9736**	.9527

TABLE 7
Comparison showing improvement of proposed method over existing techniques on REUTERS

Clusters	Entropy (lower is better)					Purity (higher is better)				
	CFAN [33]	SNMF-PCA [35]	TLLMC	TSSC	TLLR	CFAN [33]	SNMF-PCA [35]	TLLMC	TSSC	TLLR
2	.0651	.0651	.0651	**.0493**	.0651	.9912	.9912	**.9912**	.9735	**.9912**
4	.3751	.3700	.3257	**.2103**	.2508	.7934	.8035	.8618	**.8984**	.8061
6	.2029	.2521	.3268	**.1905**	.2435	.8719	.8380	.8040	**.8855**	.8595
8	.2258	.2829	.1646	**.2811**	.2279	.8586	.8076	.9252	**.9135**	.8524
10	.3677	.4464	.3166	**.2579**	.3286	.6690	.6288	.7586	**.8069**	.7331
Avg.	.2360	.2701	.2213	**.1978**	.2334	.8502	.8260	.8777	**.8950**	.8605

TABLE 8
Comparison showing improvement of proposed method over existing techniques on NewsGroup

Clusters	Entropy (lower is better)					Purity (higher is better)				
	CFAN [33]	SNMF-PCA [35]	TLLMC	TSSC	TLLR	CFAN [33]	SNMF-PCA [35]	TLLMC	TSSC	TLLR
2	.8556	.9843	.8841	**.8172**	.8131	.6867	.5500	.6500	**.7233**	**.7233**
4	.6065	.7301	.6114	**.5911**	.5755	.6083	.5183	.6540	**.6567**	.6575
6	.5441	.6492	.4835	**.4697**	.5711	.6017	.5322	.6806	**.7050**	.6028
8	.5505	.6562	.4989	**.4673**	.5942	.5708	.4700	.6338	**.6721**	.5259
10	.5395	.6025	.4602	**.4449**	.5399	.5543	.4977	.6433	**.6690**	.5600
Avg.	.6037	.7010	.5664	**.5490**	.6120	.6002	.5176	.6554	**.6793**	.6013

The results show that only for the easiest dataset (TDT2), CFAN yields slightly better results than some of our proposed variants. Note that, by easiest we mean the dataset where all the algorithms yield good results. But even for this dataset the best result is obtained from TSSC. For the more difficult datasets, all our proposed methods excel over the ones compared against; that too by a significant margin.

5 Conclusion

In this work, we have incorporated subspace clustering formulations into the transform learning framework. This results in three variants – transformed locally linear manifold clustering, transformed sparse subspace clustering and transformed low rank representation. We have also propose kernelized versions of the aforesaid three variants.

Experiments have been carried out on two benchmark problems – image and document clustering. For each problem, state-of-the-art techniques are compared against. In all cases, our proposed method excels over these in terms of the metrics used here.

Acknowledgement

This work is supported by the Infosys Center for Artificial Intelligence @ IIIT Delhi and by the Indo-French CEFIPRA grant DST-CNRS-2016-02.

References

[1] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, pp.100-108, 1979.

[2] I. S. Dhillon, Y. Guan and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," ACM KDD, pp. 551-556, 2004.

[3] A. Y. Ng, M. I. Jordan and Y. Weiss, "On spectral clustering: Analysis and an algorithm," NIPS, pp. 849-856, 2002.

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

[5] N. Zhao, L. Zhang, B. Du, Q. Zhang, J. You and D. Tao, "Robust Dual Clustering with Adaptive Manifold Regularization," IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 11, pp. 2498-2509, Nov. 1 2017.

[6] Y. Chen, L. Wang and M. Dong, "Non-Negative Matrix Factorization for Semisupervised Heterogeneous Data Coclustering," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1459-1474, Oct. 2010.

[7] H. Liu, G. Yang, Z. Wu and D. Cai, "Constrained Concept Factorization for Image Representation," IEEE Transactions on Cybernetics, vol. 44, no. 7, pp. 1214-1224, July 2014.

[8] V. M. Patel, H. V. Nguyen and R. Vidal, "Latent Space Sparse Subspace Clustering," IEEE ICCV, pp. 225-232, 2013.

[9] V. M. Patel, H. V. Nguyen and R. Vidal, "Latent Space Sparse and Low-Rank Subspace Clustering," IEEE Journal of Selected Topics in Signal Processing, vol. 9, no. 4, pp. 691-701, 2015.

[10] S. Ravishankar and Y. Bresler, "Learning Sparsifying Transforms," in IEEE Transactions on Signal Processing, vol. 61, no. 5, pp. 1072-1086, 2013.

[11] A. Goh and R. Vidal, "Segmenting Motions of Different Types by Unsupervised Manifold Clustering," IEEE CVPR, pp. 1-6, 2007.

[12] T. Blumensath, "Directional Clustering Through Matrix Factorization," IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 10, pp. 2095-2107, Oct. 2016.

[13] E. Elhamifar and R. Vidal, "Sparse Subspace Clustering: Algorithm, Theory, and Applications," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 11, pp. 2765-2781, 2013.

[14] M. Tschannen and H. Bölcskei, "Noisy Subspace Clustering via Matching Pursuits," IEEE Transactions on Information Theory, vol. PP, no. 99, pp. 1-1.

[15] Y. Chen, G. Li and Y. Gu, "Active Orthogonal Matching Pursuit for Sparse Subspace Clustering," IEEE Signal Processing Letters, vol. 25, no. 2, pp. 164-168, 2018.

[16] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu and Y. 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.

[17] R. He, L. Wang, Z. Sun, Y. Zhang and B. Li, "Information Theoretic Subspace Clustering," IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 12, pp. 2643-2655, 2016.

[18] J. Shi and J. Malik, "Normalized cuts and image segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 888-905, 2000.

[19] A. C. Türkmen, "A Review of Nonnegative Matrix Factorization Methods for Clustering", arXiv:1507.03194v2.

[20] S. Ravishankar and Y. Bresler, "Closed-form solutions within sparsifying transform learning," IEEE ICASSP, pp. 5378-5382, 2013.

[21] S. Ravishankar and Y. Bresler, "Online Sparsifying Transform Learning—Part II: Convergence Analysis," IEEE Journal of Selected Topics in Signal Processing, vol. 9, no. 4, pp. 637-646, 2015.

[22] M. Yang, L. Zhang, X. Feng and D. Zhang, "Sparse representation based fisher discrimination dictionary learning for image classification," International Journal of Computer Vision, vol. 109, no. 3, pp. 209-232, 2014.

[23] Z. Jiang, Z. Lin and L. S. Davis, "Label Consistent K-SVD: Learning a Discriminative Dictionary for Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 11, pp. 2651-2664, 2013.

[24] J. Guo, Y. Guo, X. Kong, M. Zhang and R. He, "Discriminative Analysis Dictionary Learning," AAAI, pp. 1617-1623, 2016.

[25] X. Peng, S. Xiao, J. Feng, W. Y. Yau and Z. Yi, "Deep Subspace Clustering with Sparsity Prior," IJCAI, pp. 1925-1931, 2016.

[26] F. Tian, B. Gao, Q. Cui, E. Chen and T. Y. Liu, "Learning deep representations for graph clustering," AAAI, pp. 1293-1299, 2014.

[27] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization", Journal of Optimization Theory and Applications, vol. 109, no. 3, pp. 475-494, 2001.

[28] I. Daubechies, M. Defrise and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," Communications on pure and applied mathematics, vol. 57, no. 11, pp.1413-1457, 2004.

[29] A. Majumdar and R. K. Ward, "Some Empirical Advances in Matrix Completion", Signal Processing, vol. 91, no. 5, pp. 1334-1338, 2011.

[30] J. Maggu and A. Majumdar, "Kernel transform learning," Pattern Recognition Letters, vol. 98, pp.117-122, 2017.

[31] www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[32] https://computervisiononline.com/dataset/1105138686

[33] X. Pei, C. Chen and W. Gong, "Concept Factorization With Adaptive Neighbors for Document Clustering," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 2, pp. 343-352, 2018.

[34] http://www.cad.zju.edu.cn/home/dengcai/Data/data.html

[35] K. Allab, L. Labiod and M. Nadif, "A Semi-NMF-PCA Unified Framework for Data Clustering," IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 1, pp. 2-16, Jan. 1 2017.