Signal Processing 174 (2020) 107628

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Short communication

Deeply transformed subspace clustering

Jyoti Maggu[a], Angshul Majumdar[a,*], Emilie Chouzenoux[b], Giovanni Chierchia[c]

[a] *Indraprastha Institute of Technology, New Delhi, India*
[b] *Université Paris-Saclay, CentraleSupélec, Inria, CVN, Gif-sur-Yvette, France*
[c] *University of Paris-Est, LIGM, UMR, CNRS 8019, France*

A R T I C L E I N F O

Article history:
Received 26 May 2019
Revised 25 February 2020
Accepted 24 April 2020
Available online 12 May 2020

Keywords:
Clustering
Deep learning
Dictionary learning

A B S T R A C T

Subspace clustering assumes that the data is separable into separate subspaces; this assumption may not always hold. For such cases, we assume that, even if the raw data is not separable into subspaces, one can learn a deep representation such that the learnt representation is separable into subspaces. To achieve the intended goal, we propose to embed subspace clustering techniques (locally linear manifold clustering, sparse subspace clustering and low rank representation) into deep transform learning. The entire formulation is jointly learnt; giving rise to a new class of methods called deeply transformed subspace clustering (DTSC). To test the performance of the proposed techniques, benchmarking is performed on image clustering problems. Comparison with state-of-the-art clustering techniques shows that our formulation improves upon them.

1. Introduction

Clustering has been a classical problem in machine learning. It studies how signals are naturally grouped together. Perhaps the simplest and most widely used clustering technique is the K-means [1]. It gathers samples such that the total (Euclidean) distance of the data points within the clusters is minimized. The problem is NP hard, and hence is usually solved greedily. One of the limitations of K-means is that it may fail to capture non-linear relationships. The simple way to overcome this issue is the kernel K-means [2]. The concept remains the same as in any kernel trick: the standard Euclidean distances between the samples is replaced by a kernelized version of it, within the K-means framework. Related to the kernel K-means is spectral clustering [2,3]; the later generalizes over the former by replacing kernelized data matrix to any similarity measure (not restricting to Mercer kernels) called the ‘affinity matrix’. Subspace clustering techniques [4,5] are based on a slightly different model that assumes that the samples from the same cluster will lie in the same subspace. Finding the clusters boils down to finding the different subspaces. There can be different ways to find the subspaces – these will be discussed later.

Subspace clustering operates on the raw data; the data might be such that it is not separable into subspaces in the original domain. In such a case, one may be able to learn an alternative representation space, such that the data is separable after being projected in this new domain. Several studies in the past have relied on this strategy. For example, in [6,7] a tight-frame was learnt from the data along with the subspace clustering formulation; even though the original data was not separable into subspaces, the projection within the learnt tight-frame made it separable. In [8,9] representations were learnt via autoencoders and fed into clustering algorithms (K-means in [8] and subspace clustering in [9]). The assumptions in these studies remained the same, i.e. even if the original data is not separable into subspaces, their corresponding representations will be. The difference between [6,7] and [8,9] is that the former are shallow techniques while the later are based on deep learning.

Our work is based on similar assumptions. We assume that even if the data is not separable into subspaces in the original space, we can learn deep representation from the data such that it is tailored to be separable in the representation space. Our preliminary work on this topic can be found in [10]; we showed that by learning a single layer of transform jointly with the locally linear manifold clustering, better results can be obtained. It was a shallow technique involving only one layer of transform learning. Our proposed work is an extension of this basic approach.

- The first novelty is to learn deeper representations via deep transform learning (DTL) [11].
- The second novelty is to incorporate three variants of subspace clustering – i) Locally linear manifold clustering (LLMC), ii) sparse subspace clustering (SSC), and iii) low rank representation (LRR) into the DTL model.

* Corresponding author.
E-mail addresses: jyotim@iiitd.ac.in (J. Maggu), angshulm@iiitd.ac.in, angshul@iiitd.ac.in, angshulm@ece.ubc.ca, angshulm@ece.ubc.ca (A. Majumdar), emilie.chouzenoux@centralesupelec.fr (E. Chouzenoux), giovanni.chierchia@esiee.fr (G. Chierchia).

https://doi.org/10.1016/j.sigpro.2020.107628
0165-1684/© 2020 Elsevier B.V. All rights reserved.

Operationally, we embed the clustering formulation into the deep transform learning paradigm, and jointly solve it using variable splitting optimization method. The assumption here is that, even if the original data is not separable into subspaces, by non-linearly transforming the data (via DTL), we will learn a representation which will be separable into subspaces.

The rest of the paper will be organized into the following sections. Relevant background for our work is discussed in Section 2. The proposed approach is detailed in Section 3. The experimental results are shown in Section 4. Conclusions of this work are discussed in Section 5.

2. Background

2.1. Subspace clustering

Subspace clustering assumes that the data is naturally segregated into subspaces. To find these clusters, the first task is to identify these subspaces. In the case of locally linear manifold clustering (LLMC) [12], sparse subspace clustering (SSC) [13] or low rank representation (LRR) [14], the general idea remains the same: in order to identify the subspaces, each sample is expressed as a linear combination of other samples, as follows:

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

Here x_i ($\in \mathbb{R}^m$) denotes the ith sample and $X_{i^c}(\in \mathbb{R}^{m\times n-1})$ all other samples; c_i ($\in \mathbb{R}^{n-1}$) is the corresponding linear weight vector. The information about the subspaces is embedded in the coefficients c_i. The coefficients are solved by the following,

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

Hereabove, R denotes the regularization term. Three main regularization choices have been used in the literature. For LLMC, there is no regularization, i.e. $R = 0$. For SSC, R is a sparsity promoting l_1-norm. For LRR, R is a low-rank penalty usually taking the form of nuclear norm. The first term in (2), i.e. the Euclidean norm comes from (1).

Once the coefficients are estimated, the next step is to segment (i.e. cluster) the data. This requires the application of spectral clustering. For that, an affinity matrix is computed from the coefficient matrix $C = [\breve{c}_1|\ldots|\breve{c}_n]$ obtained from all n samples. Note that $\breve{c}_i(\in \mathbb{R}^n)$ is defined from $c_i(\in \mathbb{R}^{n-1})$ by putting zero in the ith position. There is no unique definition for the affinity matrix; the only requirement is that it needs to be symmetric. Several variants have been proposed for constructing it from C. The most commonly used is probably the following:

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

Once the affinity matrix (A) is defined (by using a suitable formula), the third step is to segment the clusters. A spectral clustering algorithm applied on A (eg, Normalized-Cuts on A) is used for this purpose.

2.2. Transformed subspace clustering

The basic formulation for transformed subspace clustering was proposed by the authors in [10]. Instead of learning the clusters from the raw data, they are learnt from the transform coefficients. The learning proceeded in a joint fashion. The complete formulation is given as follows:

$$\underbrace{\min_{T,Z,C} \|TX - Z\|_F^2 + \lambda\left(\|T\|_F^2 - \log\det T\right) + \mu\|Z\|_1}_{\text{Transform Learning}}$$
$$\underbrace{+\gamma \sum_i \|z_i - Z_{i^c} c_i\|_2^2 + R(C)}_{\text{Subspace Clustering}} \tag{4}$$

We will explain each of the terms (Transform Learning and Subspace Clustering) separately. Note that in [10], the LLMC formulation was used so there was no regularization term. The first portion of the formulation (4) corresponds to that of transform learning [15]. It is the analysis equivalent of dictionary learning; transform learning analyses the data by learning a transform / basis to produce coefficients. Mathematically this is expressed as,

$$TX = Z \tag{5}$$

Here T is the transform, X is the data and Z the corresponding coefficients. Learning then proceeds by solving the following optimization problem (i.e. the first part of (4)):

$$\min_{T,Z} \|TX - Z\|_F^2 + \lambda\left(\|T\|_F^2 - \log\det T\right) + \mu\|Z\|_1 \tag{6}$$

The first term (Euclidean norm) in (6) comes from (5). The second term is the regularization on (T) and the third term the regularization on (Z). Here, one can have a trivial solution $T = 0$ and $Z = 0$. In order to prevent this $-\log\det T$ is introduced. However, this term will favor a degenerate solution where $T \to \infty$ and $Z \to 0$. To prevent the degenerate solution $\|T\|_F^2$ is introduced. The l_1-norm on Z enforces sparsity.

In practice, alternating minimization is used, that is T and Z are updated alternatingly until convergence [16]. The update for Z has a closed form, namely a soft thresholding operator; the closed form update for T can be found for instance in [17].

The solution for our prior formulation (4) proceeds via alternating between the three variables T, Z and C. Although the initial work [10] did not use any regularization, one can easily extend it to incorporate SSC and LRR formulations.

The idea behind the aforementioned approach was motivated by [6–9], i.e. even if the raw data X is not separable into subspaces, the transform coefficients (Z) will lie in different subspaces. However, the learnt transform (T) is linear. Therefore, the projection it may learn is mostly as good as the raw data as far as separability into subspaces is concerned.

2.3. Deep clustering

So far, up to our knowledge, there has been only one single work that incorporates subspace clustering into a deep learning framework. In [9], they incorporate the sparse subspace clustering formulation into the features from the deepest layer of a stacked autoencoder. Mathematically the formulation is as follows,

$$\min_{W'_i, W_i, C} \underbrace{\left\|X - W'_1\varphi\left(W'_2\varphi\left(W'_3\varphi(W_3\varphi(W_2\varphi(W_1X)))\right)\right)\right\|_F^2}_{\text{Stacked Autoencoder}}$$
$$\underbrace{+\gamma \sum_i \|(W_3\varphi(W_2\varphi(W_1X)))_i - (W_3\varphi(W_2\varphi(W_1X)))_{i^c} c_i\|_2^2 + \lambda\|c_i\|_1}_{\text{Subspace Clustering}} \tag{7}$$

Here the W_3, W_2 and W_1 represent the encoder weights for a three layer autoencoder and W'_3, W'_2 and W'_1 corresponds to the decoder weights. The term $W_3\varphi(W_2\varphi(W_1X))$ represents the autoencoder features at the deepest layer, on which spectral clustering is applied.

Here, we show the formulation for three layers. W'_i denotes the i^th level of decoder and W_i denotes the i^th level of decoder. The l_1-norm on c_i aims at promoting sparsity. The idea is like that of transformed subspace clustering; the clustering formulation is incorporated into the representation learning model.

There have been other clustering formulations embedded into the deep learning framework. In [18], a naïve solution was considered, where stacked autoencoders are used for representation learning followed by separate K-means / spectral clustering. A more sophisticated version of it [19], jointly learns the representation from a stacked autoencoder while incorporating K-means clustering formulation. A slightly different version was proposed in [20], where instead of using the standard Euclidean distance as the measure for K-means clustering, the Kullback-Leibler divergence was used. In [21], the encoder-decoder structure was replaced by a simple neural network trained to have orthogonal outputs, thus yielding a deep version of spectral clustering.

In all the aforesaid formulations, the underlying assumption remains the same. The deep network learns non-linear projections such that the learnt representation is separable into subspaces even when the original data is not. There is another class of deep learning techniques which are deeper extensions of the famous non-negative matrix factorization (NMF) based clustering framework. We can mention for instance [22] on face recognition, which argues that representations from each layer form clusters based on different attributes, e.g. if one is clustering faces, one layer may cluster based on gender, another layer may cluster based on ethnicity etc.

3. Deeply transformed subspace clustering

The idea of deep transform learning has been recently introduced in our work [11]. Just like any other deep learning model, we learn deep representation by repeatedly applying transform learning on the data. Mathematically this is expressed as follows,

$$T_3T_2T_1X = Z, \text{ subject to } T_2T_1X \geq 0 \text{ and } T_1X \geq 0 \tag{8}$$

Note that we denote use arbitrary activation functions as before, but introduce specific ReLU type non-linearity as constraints.

We will show here formulations for three layers in (8) but it is worth noting that the generalization to higher number of layers is straightforward. The optimization problem is then posed as:

$$\min_{T_i's,Z} \|T_3T_2T_1X - Z\|_F^2 + \lambda \sum_{i=1}^{3} \left(\|T_i\|_F^2 - \log\det T_i\right) \\ s.t. T_1X \geq 0 \text{ and } T_2T_1X \geq 0 \tag{9}$$

All the terms in (9) have been defined before, making the hereabove expression self explanatory. Note that we have dropped the sparsity promoting term on the coefficients. Usually in deep learning, the dimensionality of the coefficients reduces in each layer so that the representation is naturally compact and sparsity is not needed.

Using an unsupervised strategy, one can in principle take the learnt representation as an input of any classifier for segmentation. However, it will be shown in this work that such a greedy piecemeal formulation usually does not perform very well. A better approach is to learn the deep representation tailored for projections. For this work, we propose to embed the subspace clustering formulation into the deep transform learning to jointly learn the representation and clustering. This is in a similar line than the prior studies [19–21], where the goal was to learn a deep projection space (using deep autoencoders) that are conducive to clustering.

We reiterate the core idea behind this work. In subspace clustering, it is assumed that the data is naturally separated into certain subspaces. This may not always hold. Here, we are assuming that even if the original data is not separable into subspaces, by learning a non-linear representation of it (via deep transform learning) tailored for clustering, the representation will fall into separate subspaces.

Mathematically our final formulation is expressed as,

$$\min_{T_1,T_2,T_3,X_2,X_3,Z,C} \|T_3T_2T_1X - Z\|_F^2 \\ +\lambda \sum_{i=1}^{3} \left(\|T_i\|_F^2 - \log\det T_i\right) + \gamma \sum_i \|z_i - Z_{i^c}c_i\|_2^2 + R(C) \\ s.t. T_1X \geq 0 \text{ and } T_2T_1X \geq 0 \tag{10}$$

This (10) is our complete formulation. The idea is to apply subspace clustering on the deepest layer of transform coefficients. To solve (10), we follow the popular variable splitting strategy. After introducing proxy variables, the resulting augmented Lagrangian function is minimized via alternating direction method of multipliers (ADMM). In the case of (10), we introduce two proxy variables $T_2T_1X = X_3$ and $T_1X = X_2$. The augmented Lagrangian function reads,

$$\min_{T_1,T_2,T_3,X_2,X_3,Z,C} \|T_3X_3 - Z\|_F^2 + \mu_1\|T_2X_2 - X_3\|_F^2 + \mu_2\|T_1X - X_2\|_F^2 \\ +\lambda \sum_{i=1}^{3} \left(\|T_i\|_F^2 - \log\det T_i\right) + \gamma \sum_i \|z_i - Z_{i^c}c_i\|_2^2 + R(C) \\ s.t. X_3 \geq 0 \text{ and } X_2 \geq 0 \tag{11}$$

In (11), the hyper-parameters μ_1 and μ_2 correspond to weights associated to the representation in shallower layers. We argue that there is no reason to prefer one layer over the other, therefore we assign μ_1=μ_2=1. With this slight simplification, we have,

$$\min_{T_1,T_2,T_3,X_2,X_3,Z,C} \|T_3X_3 - Z\|_F^2 + \|T_2X_2 - X_3\|_F^2 + \|T_1X - X_2\|_F^2 \\ +\lambda \sum_{i=1}^{3} \left(\|T_i\|_F^2 - \log\det T_i\right) + \gamma \sum_i \|z_i - Z_{i^c}c_i\|_2^2 + R(C) \\ s.t. X_3 \geq 0 \text{ and } X_2 \geq 0 \tag{12}$$

The problem (12) can now be solved using ADMM [23]. Each of the variables are updated separately by solving the following sub-problems.

$$P1 : \min_{T_1} \|T_1X - X_2\|_F^2 + \lambda(\|T_1\|_F^2 - \log\det T_1)$$
$$P2 : \min_{T_2} \|T_2X_2 - X_3\|_F^2 + \lambda(\|T_2\|_F^2 - \log\det T_2)$$
$$P3 : \min_{T_3} \|T_3X_3 - Z\|_F^2 + \lambda(\|T_3\|_F^2 - \log\det T_3)$$
$$P4 : \min_{X_3} \|T_3X_3 - Z\|_F^2 + \|T_2X_2 - X_3\|_F^2 s.t. X_3 \geq 0$$
$$P5 : \min_{X_2} \|T_2X_2 - X_3\|_F^2 + \|T_1X - X_2\|_F^2 s.t. X_2 \geq 0$$
$$P6 : \min_{Z} \|T_3X_3 - Z\|_F^2 + \gamma \sum_i \|z_i - Z_{i^c}c_i\|_2^2$$
$$P7 : \min_{C} \sum_i \|z_i - Z_{i^c}c_i\|_2^2 + R(C)$$

Sub-problems P1 to P3 are standard transform updates whose closed form solution is given in [17]. P4 and P5 are constrained least square problems that will be solved approximately using a pseudoinverse operation followed by capping all the negative values to zero. P6 is a simple least squares problem. The solution to P7 will depend on the regularization used. With no regularization (i.e., LLMC), it has a closed form update via the pseudoinverse. With l_1-norm regularization, P7 can be solved via iterative soft thresholding-based solver such as [24]; this case pertains to SSC. When the regularizer in P7 is a nuclear norm (i.e., LRR), one needs to solve it via singular value shrinkage [25].

This concludes the derivation of the main algorithm. Once (10) is solved, our work proceeds in the same fashion as standard subspace clustering. Given C, we compute the affinity matrix using (3), which is then segmented / clustered by Normalized cut.

Table 1
Comparison with benchmarks on COIL 20.

Metric	SSC	KSSC	LRR	KLRR	DSC	DKM	DMF	DTLLMC	DTSC	DTLRR
Accuracy	.79	.72	.72	.71	.85	.88	.86	**.93**	**.98**	.81
NMI	.89	.79	.84	.80	.91	.94	.92	**.97**	**.98**	.89
ARI	.76	.64	.65	.63	.84	.86	.85	**.89**	**.90**	.80
Precision	.70	.63	.65	.61	.82	.85	.84	**.88**	**.90**	.72
F-measure	.78	.65	.66	.63	.85	.87	.84	**.91**	**.93**	.81

Table 2
Comparison with benchmarks on Extended Yale B.

Metric	SSC	KSSC	LRR	KLRR	DSC	DKM	DMF	DTLLMC	DTSC	DTLRR
Accuracy	.70	.70	.71	.70	.88	.91	.89	**.96**	**.99**	.84
NMI	.83	.83	.80	.80	.90	.92	.90	**.97**	**.98**	.89
ARI	.64	.65	.63	.63	.83	.90	.83	**.95**	**.96**	.77
Precision	.65	.67	.62	.61	.79	.91	.80	**.95**	**.99**	.71
F-measure	.66	.68	.65	.65	.83	.90	.84	**.93**	**.95**	.76

4. Experimental results

In this section we compare our method with three deep clustering benchmarks, namely deep sparse subspace clustering (DSC) [9], deep K-means clustering (DKM) [19] and deep matrix factorization (DMF) [22]. The said studies have been published recently and have been favorably compared with traditional clustering techniques like matrix factorization, spectral clustering, subspace clustering, hierarchical clustering. For the sake of completeness of our study, we compare with four classical methods as well, sparse subspace clustering (SSC), kernel SSC (KSSC), low rank representation (LRR) and kernel LRR (KLRR). The RBF kernel is retained for kernel methods.

We follow the experimental protocol from [9]. Experiments were carried out on the COIL20[1] (object recognition) and Extended YaleB[2] (face recognition) datasets. The COIL20 database contains 1440 samples distributed over 20 objects, where each image takes the size of 32 × 32. The used YaleB consists of 2414 samples from 38 individuals, where each image is with size of 192 × 168. For both datasets, DSIFT (dense scale invariant feature transform) features were extracted. They were further reduced by PCA to a dimensionality of 300. Since the ground truth (class labels) for these datasets is available, clustering accuracy can be measured. Here, we use Accuracy, NMI (normalized mutual information), ARI (adjusted rank index), Precision and F-score. The results are shown in Table 1 (COIL20) and Table 2 (YaleB). Since the last stage of all the clustering algorithms involves K-means, we ran the experiments on 100 random runs and report the mean for each score.

The parametric settings for the methods compared against have been taken from the respective papers. For our proposed technique, we have kept λ=0.1 and γ=1 as these are standard values used in previous works in transform learning. TLLMC does not require specification of any other parameter. TSC has τ=0.1 as the sparsity promoting term and TLLR has τ=0.01 as the rank deficiency term. As we will see later for their deep counterpart, those algorithms seem robust to these parametric values (λ, γ and τ); changes by an order of magnitude to either side do not affect the results statistically.

The results in Tables 1 and 2 show that our proposed method with sparse subspace clustering yields the best results on an aggregate. The results from the LLMC based formulation are comparable to the existing benchmarks. Our formulation with LRR yields the worst results. But this is in tune with the observations in [9] – LRR formulation does not yield good results on these datasets. This may be because LRR is sensitive to outliers; most LRR based clustering formulations have an explicit outlier rejection term; we have not incorporated it here; we used the vanilla formulation. Perhaps this is the reason why the results are quite poor. We find that the classical techniques (SSC, KSSC, LRR and KLLR) perform worse than the rest of the techniques. This observation is in tune with prior studies.

Table 3
Comparison of run-times in seconds.

Technique	Coil 20	Yale B
SSC	11	9
KSSC	22	20
LRR	24	22
KLRR	48	50
DSC	62	61
DKM	87	83
DMF	57	54
DTLLMC	44	38
DTSSC	50	42
DTLRR	58	48

Our method requires specification of very few parameters. The values of λ=0.1 and γ=1 are used as usual settings in transform learning literature [15–17]; we did not tune it. The only parameter we tuned is the regularization term corresponding to the subspace clustering. Since LLMC does not have any regularization, no analysis could be carried out. For the LRR and SSC variants, variation of ARI is plotted in Fig. 1.

We have also added in Fig. 2 the evolution of the cost function along iterations, assessing the empirical convergence of our three different algorithms for the COIL 20 dataset. The convergence plots for the YALE B are of similar nature and hence are not shown here. We have also added the run-times for different techniques in Table 3. All the experiments were run on an Intel i7 processor with 32 GB RAM running a 64 bit Windows 10. The proposed techniques and DMF were based on Matlab, while DSC, DKM were based on Python.

From these run-times we find that our method is actually the fastest among all the deep learning formulations. Among the three variants we have proposed, DTLLMC is the fastest; this is because it does not have any regularization term that needs iterative steps. Among DTSSC and DTLRR, DTLRR is the slowest since it requires solving at each step a singular value decomposition, which is computationally expensive. The linear shallow methods are the fastest; but their kernelized versions are comparatively slower. This is be-

[1] www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.
[2] https://computervisiononline.com/dataset/1105138686.

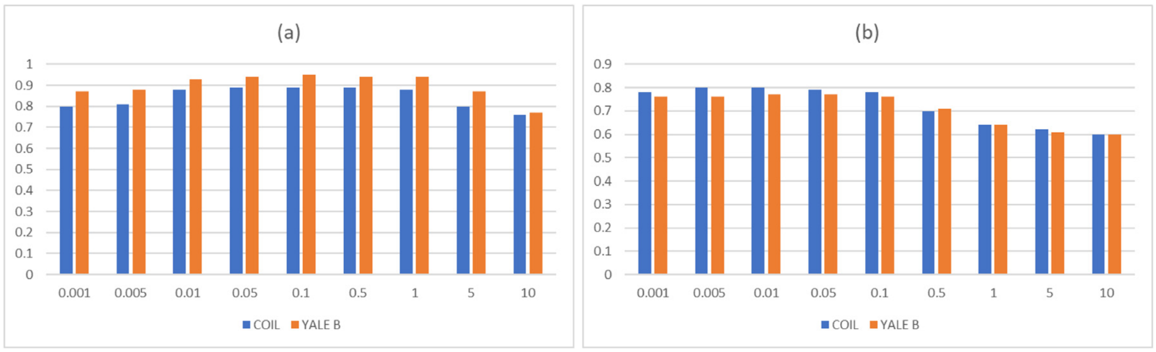

Fig. 1. Variation of DTSC (a) and DTLRR (b) with regularization value τ.

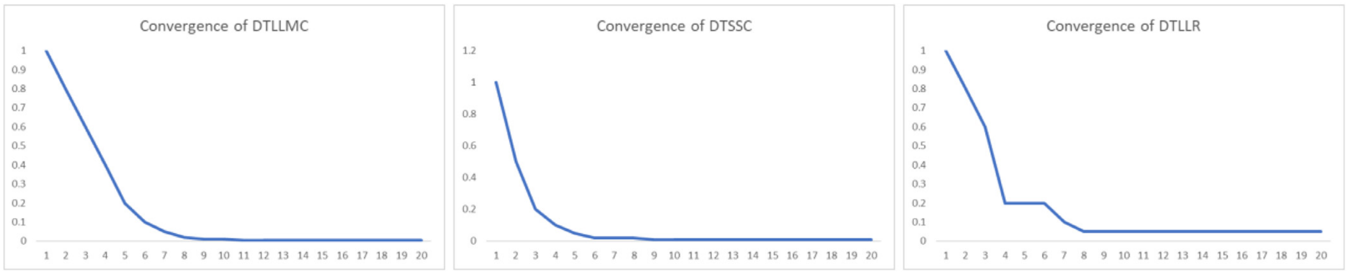

Fig. 2. Empirical Convergence Plot.

Table 4
Analysis of proposed method (DTSC) on COIL 20.

Metric	1 layer		2 layer		3 layer		4 layer	
	Greedy	Joint	Greedy	Joint	Greedy	Joint	Greedy	Joint
Accuracy	.96	.97	.96	.98.	.96	.98	.94	.99
NMI	.85	.92	.86	.94	.88	.98	.84	.95
ARI	.86	.92	.87	.94	.87	.90	.83	.96
Precision	.82	.87	.82	.88	.83	.90	.81	.88
F-measure	.81	.86	.82	.88	.82.	.93	.80	.89

Table 5
Analysis of proposed method (DTSC) on Extended YaleB.

Metric	1 layer		2 layer		3 layer		4 layer	
	Greedy	Joint	Greedy	Joint	Greedy	Joint	Greedy	Joint
Accuracy	.73	.98	.76	.99	.76	.99	.74	.99
NMI	.39	.94	.41	.94	.43	.98	.42	.96
ARI	.38	.95	.43	.95	.44	.96	.45	.95
Precision	.44	.98	.58	.99	.59	.99	.59	.99
F-measure	.42	.94	.51	.95	.53	.96	.52	.95

cause of the increased size of the kernel matrix compared to the original dimensions.

So far, we have shown the best results from our proposed method. In the next set of experiments, we show the influence of a variation in number of layers and the results obtained with the greedy solution. In the later, we learn the deep transform separately and feed the features into subspace clustering. Since the SSC model yields the best results, we are showing the results on this formulation. The results are shown in Tables 4 and 5.

We find that the results improve from one to three layers, but once we go beyond three layers the results deteriorate. This is because, with more layers the number of parameters to learn increases; with limited volume of training data (as is the case), this may lead to over-fitting and subsequent deterioration of results. Between the joint and greedy formulations, our joint formulation yields clearly better results. This is expected because this formulation learns the weights with the goal of clustering. The greedy unsupervised formulation does not present this advantage.

5. Conclusion

In this work we propose deeply transformed subspace clustering. Several layers of transforms are used to analyze the data such that the learnt representations are separable into subspaces. This stems from the assumption that even if the data is not separable into subspaces in the original space, their non-linearly learnt representations will be. In future, this approach can be extended to any clustering algorithm. For example, we can take the coefficients from the deepest layer of transform coefficients and input them to p-spectral clustering [26] or we can have a semi-supervised version as in [27]. There can be other extensions as well.

Declarations of Competing Interest

The authors declare conflict of interest with the authors of the following papers:

X. Peng, S. Xiao, J. Feng, W. Y. Yau and Z. Yi, “Deep Sub-space Clustering with Sparsity Prior,” IJCAI, pp. 1925–1931, 2016.

B. Yang, X. Fu, N. D. Sidiropoulos and M. Hong, “Towards k-means-friendly spaces: Simultaneous deep learning and clustering,” ICML, pp. 3861–3870, 2017.

G. Trigeorgis, K. Bousmalis, S. Zafeiriou and B. W. Schuller, "A Deep Matrix Factorization Method for Learning Attribute Representations," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 3, pp. 417–429, 2017.

CRediT authorship contribution statement

Jyoti Maggu: Data curation, Investigation, Validation, Methodology. **Angshul Majumdar:** Supervision, Project administration, Writing - original draft. **Emilie Chouzenoux:** Conceptualization, Writing - review & editing. **Giovanni Chierchia:** Conceptualization, Writing - review & editing.

Acknowledgment

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

References

[1] J.A. Hartigan, M.A. Wong, Algorithm AS 136: a k-means clustering algorithm, J.R. Stat. Soc. Ser. C 28 (1) (1979) 100–108 (Applied Statistics).
[2] I.S. Dhillon, Y. Guan, B. Kulis, Kernel k-means: spectral clustering and normalized cuts, ACM KDD (2004) 551–556.
[3] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, NIPS (2002) 849–856.
[4] R. Vidal, Subspace Clustering, IEEE Signal Process. Mag. 28 (2) (2011) 52–68.
[5] Y. Oktar, M. Turkan, A review of sparsity-based clustering methods, Signal Process. 148 (2018) 20–30.
[6] V.M. Patel, H.V. Nguyen, R. Vidal, Latent space sparse subspace clustering, IEEE ICCV (2013) 225–232.
[7] V.M. Patel, H.V. Nguyen, R. Vidal, Latent space sparse and low-rank subspace clustering, IEEE J. Sel. Top. Signal. Process. 9 (4) (2015) 691–701.
[8] F. Tian, B. Gao, Q. Cui, E. Chen, T.Y. Liu, Learning deep representations for graph clustering, AAAI (2014) 1293–1299.
[9] X. Peng, S. Xiao, J. Feng, W.Y. Yau, Z. Yi, Deep Sub-space Clustering with Sparsity Prior, IJCAI (2016) 1925–1931.
[10] J. Maggu, A. Majumdar, E. Chouzenoux, Transformed locally linear manifold clustering, EUSIPCO (2018) 1057–1061.
[11] J. Maggu, A. Majumdar, Unsupervised deep transform learning, IEEE ICASSP (2018) 6782–6786.
[12] A. Goh, R. Vidal, Segmenting motions of different types by unsupervised manifold clustering, IEEE CVPR (2007) 1–6.
[13] E. Elhamifar, R. Vidal, Sparse subspace clustering: algorithm, theory, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2765–2781.
[14] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, Y. Ma, Robust recovery of subspace structures by low-rank representation, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 171–184.
[15] S. Ravishankar, Y. Bresler, Learning sparsifying transforms, IEEE Trans. Signal Process. 61 (5) (2013) 1072–1086.
[16] S. Ravishankar, Y. Bresler, Online sparsifying transform learning—part II: convergence analysis, IEEE J. Sel. Top. Signal Process. 9 (4) (2015) 637–646.
[17] S. Ravishankar, Y. Bresler, Closed-form solutions within sparsifying transform learning, IEEE ICASSP (2013) 5378–5382.
[18] F. Tian, B. Gao, Q. Cui, E. Chen, T.Y. Liu, Learning deep representations for graph clustering, AAAI (2014) 1293–1299.
[19] B. Yang, X. Fu, N.D. Sidiropoulos, M. Hong, Towards k-means-friendly spaces: simultaneous deep learning and clustering, ICML (2017) 3861–3870.
[20] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, ICML (2016) 478–487.
[21] M. El Gheche, G. Chierchia, and P. Frossard. “OrthoNet: multilayer Network Data Clustering.” Preprint ArXiv:1811.00821, 2019.
[22] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, B.W. Schuller, A deep semi-NMF model for learning hidden representations, ICML (2014).
[23] M. Hong, Z.Q. Luo, M. Razaviyayn, Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems, SIAM J. Optimiz. 26 (1) (2016) 337–364.
[24] J. Zeng, S. Lin, Z. Xu, Sparse solution of underdetermined linear equations via adaptively iterative thresholding, Signal Process. 97 (2014) 152–161.
[25] X. Lin, G. Wei, Accelerated reweighted nuclear norm minimization algorithm for low rank matrix recovery, Signal Process. 114 (2015) 24–33.
[26] S. Ding, L. Cong, Q. Hua, H. Jia, Z. Shi, A multiway p-spectral clustering algorithm, Knowl. Based Syst. 164 (2019) 371–377.
[27] S. Ding, H. Jia, M. Du, Y. Xue, A semi-supervised approximate spectral clustering algorithm based on HMRF model, Inf. Sci. 429 (2018) 215–228.