Deeply Transformed Subspace Clustering

Jyoti Maggu, Angshul Majumdar, Emilie Chouzenoux, Giovanni Chierchia

Deeply Transformed Subspace Clustering

Jyoti Maggu[1], Angshul Majumdar[1], Emilie Chouzenoux[2] and Giovanni Chierchia[2]

[1]Indraprastha Institute of Information Technology, New Delhi, India

[2]University of Paris-Est, LIGM, UMR CNRS 8019, France

jyotim@iiitd.ac.in, angshulm@iiitd.ac.in, emilie.chouzenoux@univ-mlv.fr and giovanni.chierchia@esiee.fr

Abstract – 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 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 groups the 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 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 later generalizes over the former by replacing kernelized data matrix to any similarity measure (not restricting to Mercer kernels) calling it the 'affinity matrix'. Subspace clustering techniques [4, 5] are based on a slightly different model; it 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 a projection, such that the data is separable in the projected domain. Several studies in the past

have shown that indeed may be the case. 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 by 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 extension is to learn deeper representations via deep transform learning (DTL) [11].
- The second extension 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.

Operationally, we embed the clustering formulation into the deep transform learning paradigm, and jointly solve it using variable splitting. 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.

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. The general formulation, be it locally linear manifold clustering (LLMC) [12], sparse subspace clustering (SSC) [13] or low rank representation (LRR) [14], remains the same; in order to identify the subspaces each sample is expressed as a linear combination of other samples.

$$x_i = X_{i^c} c_i, \forall i \text{ in } \{1,...,n\} \quad (1)$$

Here x_i ($\in \mathbb{R}^m$) denotes the i^{th} 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,

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

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. For LRR, it is a low-rank penalty usually in the form of nuclear norm.

Once the coefficients are estimated, the next step is to segment the data. This requires application of spectral clustering. For that, the affinity matrix is defined from the coefficient matrix $C = [\breve{c}_1 | ... | \breve{c}_n]$, 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 i^{th} 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 for constructing it from C; but the most commonly used is the following –

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

Once the affinity matrix is defined (by using any suitable formula), the third step is to segment the clusters. Usually spectral clustering algorithm (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 –

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

Note that in [10], the LLMC formulation was used so there was no regularization. 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 proceeds by solving the following optimization problem – first part of (4).

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

T and Z are updated alternatingly till the solution converges [16]. The update for Z has a closed form – soft thresholding; the closed form update for T has also been given 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 this approach is 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 learns is mostly as good as the raw data as far as separability into subspaces is concerned.

2.3 Deep Clustering

So far, there has been only a 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,

$$\min_{W_i', W_i, C} \underbrace{\left\|X - W_1'\varphi\left(W_2'\varphi\left(W_3'\varphi\left(W_3\varphi\left(W_2\varphi\left(W_1X\right)\right)\right)\right)\right)\right\|_F^2}_{\text{Stacked Autoencoder}} + \underbrace{\gamma\sum_i \left\|\left(W_3\varphi\left(W_2\varphi\left(W_1X\right)\right)\right)_i - \left(W_3\varphi\left(W_2\varphi\left(W_1X\right)\right)\right)_{i^c} c_i\right\|_2^2 + \lambda\|c_i\|_1}_{\text{Subspace Clustering}} \tag{7}$$

Here are showing 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 promotes sparsity. The idea is similar to 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 where stacked autoencoders are used for representation learning followed by separate K-means / spectral clustering was proposed. 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 of 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 well known non-negative matrix factorization (NNMF) based clustering framework. In [22] this is extended to deeper layers. They argue 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 Sub-space Clustering

The idea of deep transform learning has been recently introduced [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 –

$$T_3 T_2 T_1 X = Z \tag{8}$$

Although we have not explicitly shown any activation function between the layers, we will implicitly impose rectified linear unit (ReLU) type activation. We have shown the formulation for three layer. The generalization to more number of layers will be straightforward. The optimization problem is posed as –

$$\begin{aligned} &\min_{T_i\text{'s},Z} \left\| T_3 T_2 T_1 X - Z \right\|_F^2 + \lambda \sum_{i=1}^{3} \left(\left\| T_i \right\|_F^2 - \log\det T_i \right) \\ &\textit{s.t. } T_1 X \geq 0 \text{ and } T_2 T_1 X \geq 0 \end{aligned} \tag{9}$$

Note that we have dropped the sparsity promoting term on the coefficients. Usually in deep learning, the dimensionality of the coefficients reduce in each layer; therefore the representation is naturally compact.

Using the unsupervised formulation, one can in principle take the learnt representation and put into 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, we embed the subspace clustering formulation into the deep transform learning to jointly learn the representation and clustering. The notion is similar to prior studies [19-21], where the goal is to learn a deep projections (with 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 formulation can be expressed as,

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

To solve (10), we follow the popular variable splitting strategy. After introducing the proxy variable, the augmented Lagrangian is solved via alternating direction method of multipliers (ADMM). In our case (10) we introduce two proxy variables $T_2T_1X = X_3$ and $T_1X = X_2$. The augmented Lagrangian becomes,

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

In (11), the hyper-parameters μ_1 and μ_2 correspond to the representation in shallower layers. We argue that there is no reason to prefer one layer over the other, therefore we assign $\mu_1=\mu_2=1$. With this slight simplification, we have –

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

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

$$\text{P1}: \min_{T_1} \left\|T_1X - X_2\right\|_F^2 + \lambda\left(\left\|T_1\right\|_F^2 - \log\det T_1\right)$$

$$\text{P2}: \min_{T_2} \left\|T_2X_2 - X_3\right\|_F^2 + \lambda\left(\left\|T_2\right\|_F^2 - \log\det T_2\right)$$

$$\text{P3}: \min_{T_3} \left\|T_3X_3 - Z\right\|_F^2 + \lambda\left(\left\|T_3\right\|_F^2 - \log\det T_3\right)$$

$$\text{P4}: \min_{X_3} \left\| T_3 X_3 - Z \right\|_F^2 + \left\| T_2 X_2 - X_3 \right\|_F^2 \; s.t. \; X_3 \geq 0$$

$$\text{P5}: \min_{X_2} \left\| T_2 X_2 - X_3 \right\|_F^2 + \left\| T_1 X - X_2 \right\|_F^2 \; s.t. \; X_2 \geq 0$$

$$\text{P6}: \min_{Z} \left\| T_3 X_3 - Z \right\|_F^2 + \gamma \sum_i \left\| z_i - Z_{i^c} c_i \right\|_2^2$$

$$\text{P7}: \min_{C} \sum_i \left\| z_i - Z_{i^c} c_i \right\|_2^2 + R(C)$$

Sub-problems P1 to P3 are standard transform updates whose closed form solution is given in [17]. P4 and P5 are least square problems with closed form updates – one first needs computing the pseudoinverse followed by imputing 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 (locally linear manifold clustering), it will have a closed form update via the pseudoinverse. With l_1-norm regularization P7 will have to solved via some kind of iterative soft thresholding such as [24]; this case pertains to sparse subspace clustering. When the regularizer in P7 is a nuclear norm, 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.

4. Experimental Results

In this section we compare our method with three deep clustering benchmarks – 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 compared with traditional clustering techniques like matrix factorization, spectral clustering, subspace clustering, hierarchical clustering etc. However for the sake of completeness, we compare with four classical methods as well – sparse subspace clustering (SSC), kernel SSC (KSSC), low rank representation (LRR) and kernel LRR (KLRR); the kernel methods use RBF kernel.

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 1,440 samples distributed over 20 objects, where each image is with the size of 32×32. The used YaleB consists of 2,414 samples from 38 individuals, where each image is with size of 192×168. For both the 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 are available, clustering accuracy was measured in terms of 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 involve K-means, we ran the experiments 100 times and report the mean.

The parametric settings for the methods compared against have been taken from the respective papers. For our proposed technique, we have kept $\lambda=0.1$ and $\gamma=1$; these are the standard values used in transform learning. TLLMC does not require specification of any other parameter. TSC has $\tau=0.1$ as the sparsity promoting term and TLLR has $\tau=0.01$ as the rank deficiency term. The algorithms are robust to these parametric values; changes by an order of magnitude to either side do not affect the results statistically.

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 YaleB

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

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

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

The results 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, the results are 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.

Our method requires specification of very few parameters. The values of $\lambda=0.1$ and $\gamma=1$ have been used from 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.

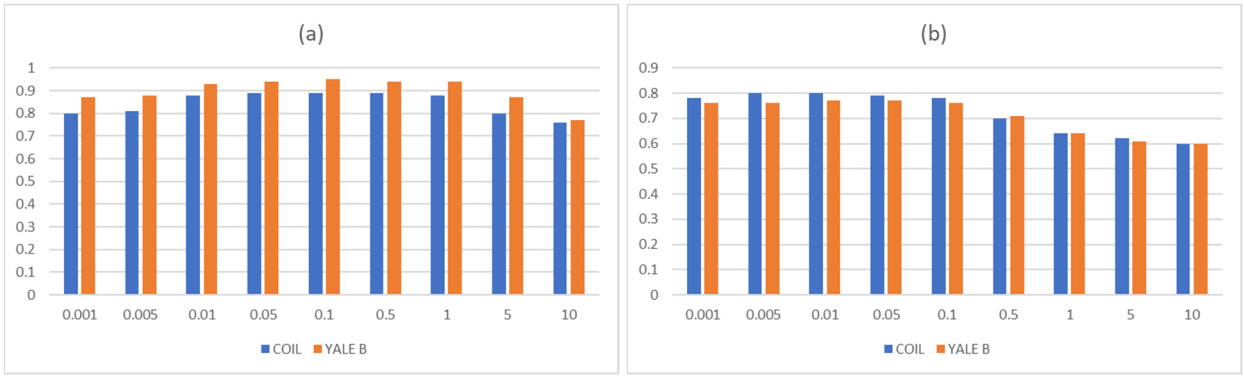

Fig. 1. Variation of DTSC (a) and DTLRR (b) with regularization value τ

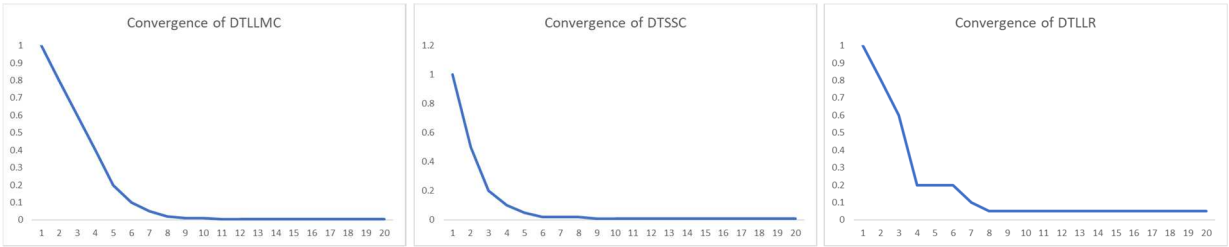

Fig. 2. Empirical Convergence Plot

We have also added empirical results on the convergence of our three different algorithms for the COIL 20 dataset. The results for the YALE B are of similar nature and hence are not shown here. We have also added the run-times for different techniques. The results are shown 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 was based on Matlab; DSC, DKM were based on Python.

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

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. DTSSC and DTLRR are slower that DTLLMC owing to the l_1-norm and nuclear norm regularizations. Among these two, DTLRR is the slowest since it requires solving a singular value decomposition, which is a computationally expensive step. The linear shallow methods are the fastest; but their kernelized versions are comparatively slower. This is because of the increases 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 results of variation in number of layers and the results of joint versus greedy solution. In the greedy solution, we learn the deep transform separately and feed the features into subspace clustering. Since the sparse subspace clustering yields the best results, we are showing the results on this formulation. The results are shown in Tables 4 and 5.

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

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 leads to over-fitting and subsequent deterioration of results. Between the joint and greedy formulations, the joint formulation yields better results. This is expected, because this formulation learns the weights with the goal of clustering. The greedy unsupervised formulation does not have this advantage.

5. Conclusion

In this work we propose deeply transformed subspace clustering. Several layers of transforms are used to analyze the data with 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. We have compared our proposed technique on two benchmark clustering datasets and compared with three contemporary deep clustering techniques. Our method shows the best results.

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.

Reference

[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] Y. Oktar, and M. Turkan, "A review of sparsity-based clustering methods," Signal Processing, vol. 148, pp. 20–30, 2018.

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

[7] 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.

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

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

[10] J. Maggu, A. Majumdar and E. Chouzenoux, "Transformed Locally Linear Manifold Clustering," EUSIPCO, pp. 1057-1061, 2018.

[11] J. Maggu and A. Majumdar, "Unsupervised Deep Transform Learning," IEEE ICASSP, pp. 6782-6786, 2018.

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

[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] 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.

[15] S. Ravishankar and Y. Bresler, "Learning Sparsifying Transforms," IEEE Transactions on Signal Processing, vol. 61, no. 5, pp. 1072-1086, March1, 2013.

[16] 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.

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

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

[19] 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.

[20] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," ICML, pp. 478-487, 2016.

[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 and B. W. Schuller, " A Deep Semi-NMF Model for Learning Hidden Representations," ICML, 2014.

[23] M. Hong, Z. Q. Luo and M. Razaviyayn, “Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems,” SIAM Journal on Optimization, vol. 26, no. 1, pp.337-364, 2016.

[24] J. Zeng, S. Lin and Z. Xu, “Sparse solution of underdetermined linear equations via adaptively iterative thresholding,” Signal Processing, vol. 97, pp. 152-161, 2014.

[25] X. Lin and G. Wei, “Accelerated reweighted nuclear norm minimization algorithm for low rank matrix recovery,” Signal Processing, vol. 114, pp. 24-33, 2015.