THE UNIVERSITY OF SYDNEY

BUSS7002

BUSINESS DISSERTATION

LSTM for Evolutionary Self-Expressive Subspace Clustering

* Another version of this thesis will be submitted to IEEE Transactions on Neural Networks and Learning Systems.

Author:
Di XU

Supervisor:
Prof. Junbin GAO

May 30, 2019

LSTM for Evolutionary Self-Expressive Subspace Clustering

Di Xu*

Abstract—**Massive volumes of high-dimensional data that evolves over time is continuously collected by contemporary information processing systems, which brings up the problem of organizing this data into clusters, i.e. achieve the purpose of dimensional deduction, and meanwhile learning its temporal evolution patterns. In this paper, a framework for evolutionary subspace clustering, referred to as LSTM-ESCM, is introduced, which aims at clustering a set of evolving high-dimensional data points that lie in a union of low-dimensional evolving subspaces. In order to obtain the parsimonious data representation at each time step, we propose to exploit the so-called self-expressive trait of the data at each time point. At the same time, LSTM networks is implemented to extract the inherited temporal patterns behind data in overall time frame. An efficient algorothm has been proposed based on MATLAB. Next, our experiment is carried out on a real-world motion segmentation problem to demonstrate the effectiveness of the proposed approach. And the results show that our suggested algorithm dramatically outperforms other known similar approaches in terms of both run time and accuracy.**

Index Terms—**subspace clustering, evolutionary clustering, self-expressive models, temporal data, real-time clustering, motion segmentation, deep learning, LSTM.**

I. Introduction

The recent decade has witnessed a gigantic explosion of data availability from various modalities and sources due to the advance of contemporary information processing systems. For instance, billions of cameras get installed worldwide and are ceaselessly generating data. This has promoted remarkable progresses and meanwhile created new challenges on how to acquire, compress, store, transmit, and process massive amounts of high-dimensional complex data. High dimensionalities of data will bring in severe computational as well as memorial burdens and mostly cut down the performance of existing algorithms. An important unsupervised learning problem encountered in such settings deals with finding informative parsimonious structures characterizing large-scale high-dimensional datasets. This task is critical for the detection of meaningful patterns in complex data and enabling accurate and efficient clustering.

Based on the observation that though the collected dataset is in high dimension, the intrinsic dimension is commonly much smaller than that of its ambient space. In computer vision, for example, the number of pixels in an image could be rather large, yet most computer vision models use only a few parameters to describe the appearance, geometry, and dynamic of a scene. And this fact has motivated the development of lots of techniques for finding a low-dimensional representation of a high-dimensional dataset for the purpose of detecting underlying data characteristics. Traditional approaches, e.g. principle component analysis (PCA), are under the assumption that data are sampled from one single low-dimensional subspace of a high-dimensional space. In reality, the data points could be yet drawn from a number of subspaces and the membership of the data points to the subspaces is probably unknown. For example, a video sequence might have several moving objects, and different subspaces might be needed to describe the motions of their corresponding objects in the scene. Thus, we need to concurrently cluster the data into several subspaces as well as detect out a low-dimensional subspace fitting each group of points. This problem has found a wide range of applications in motion segmentation and face clustering in computer vision [1], [2], image representation and compression in image processing [3], [4], hybrid system identification in systems theory [5], robust principal component analysis (RPCA) [6], and robust subspace recovery and tracking [7]–[11].

In the aforementioned settings, apart from the property that data could be reckoned as a bunch of points lying in a union of low-dimensional subspaces, data is also acquired at multiple points in time mostly. Thus, exploring its inherent temporal behaviors will obtain more information and higher clustering accuracy. For instance, we all know that feature point trajectories are related to motions in a video lying in an affine subspace [12]. Motions in any given short time interval is associated with motions in their latest past. Thus, aside from the union-of-subspaces structure in a video data, the underlying evolutionary structure is as well the key to characterize motions. As a result, designing proper frameworks that are capable of exploiting both the union-of-subspaces and temporal smoothness structures so as to conduct fast and precise clustering, especially in real-time cases where clustering is conducted in online mode and a solution is required at each time step, is of great interest.

In recent years, folks have been engaged in using deep networks to deal with various categories of temporal data and better performance has been reported in [13]–[16]. Among the existing neural network architectures, recurrent neural networks (RNNs), especially LSTM, have been proved to have even more superior performance than those of other networks and, not to mention, the commonly used traditional models. Meanwhile, conducting unsupervised clustering with LSTM structure has also been explored in diverse settings, See [17]–[19] for details.

Nonetheless, till now, few neural-networks (NN) related algorithms have covered the topic of evolutionary subspace clustering. In the meantime, most researchers focus on exploring and improving the schemes of static subspace clustering [20] with solely a few studies taking advantage of the intrinsic temporal patterns within data. Recently, [21] investigates

* Project supervisor Junbin Gao will be the second author of this thesis when submitted to IEEE Transactions on Neural Networks and Learning Systems.

both the evolutionary and a-union-of-subspaces properties of a motion segmentation problem. They introduce a convex evolutionary self-expressive model (CESM), an optimization framework that exploits the self-expressiveness property of data and learns sparse representations while taking into account prior representations. However, their processing of temporal patterns in data only limits in implementing classic weighted average smoothing techniques, which lead to sub-optimal efficiency in extracting complex information of data evolution. Thus, in this paper, we further explore the idea of CESM and extend their design by utilizing LSTM networks to learn the evolutionary self-expressive data representation. Our framework alternates between two steps - LSTM clustering and spectral clustering - to study both the data evolution and segmentation. The advantages of the proposed scheme is in two folds; not only we implement deep networks in seeking the temporal patterns as well as subspace representation of data, but we obtain as well a notably faster and more accurate algorithm compared to the past models.

The rest of this paper is organized as follows. Section II overviews existing approaches to subspace clustering, evolutionary clustering, and evolutionary subspace clustering. Section III proposes our LSTM evolutionary self-expressive subspace clustering framework. Section IV describes the implementation of this scheme in MATLAB programming. Section V presents experimental results on a motion segmentation problem. Then, the concluding remarks are stated in section VI.

II. Background

In this section, we first state the notation used all through the paper. Next, we recap multiple past subspace clustering, evolutionary clustering and evolutionary subspace clustering methods. At the end, we spotlight distinctive features of the evolutionary subspace clustering framework that we will introduce in the succeeding sections.

A. Notation

Bold capital letters denote matrices such as $\mathbf{X}$ while bold lowercase letters such as $\boldsymbol{x}$ represent vectors. Sets as well as subspaces are denoted by calligraphic letters, such as $\mathscr{X}$ for a set and $\mathscr{S}$ for a subspace. N is the number of data points in a set, e.g. $\mathscr{X}$, while D is the corresponding actual dimension of that set. n is the total number of subspaces for a set $\mathscr{X}$, with $\{\mathscr{S}_i\}_{i=1}^n$. And the dimension of a subspace $\mathscr{S}$ is denoted by d. T is the total number of time step in an evolving dataset while t is a specific timestep constrained by $1 \leq t \leq T$, where $\{\mathbf{X}_t\}_{t=1}^T$. $\mathbf{I}$ is the identity matrix. Further, $\|\mathbf{X}\|_*$ denotes the nuclear norm of $\mathbf{X}$ defined as the sum of singular values of $\mathbf{X}$. diag$(\mathbf{X})$ outputs the sum of the diagonal elements in $\mathbf{X}$. tr$(\mathbf{X})$ yields the trace of the input matrix $\mathbf{X}$. vec$(\mathbf{X})$ constructs a vector by orderly stitching all the columns of $\mathbf{X}$ together. Finally, sign(x) returns the sign of its argument and ceil(x) returns the integer rounding up to its argument.

B. Subspace Clustering

Subspace clustering has been notably highlighted over the past two decades (see, e.g., [22] and the references therein). The motivation behind is to arrange data into clusters so as to find a union of subspaces that the data points are drawn from. In specific, let $\{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$ be a given set of points drawn from a unknown union of $n \geq 1$ linear or affine subspaces $\{\mathscr{S}_i\}_{i=1}^n$ of unknown dimensions $d_i =$dim$(\mathscr{S}_i)$, $0 < d_i < D$, $i = 1, ..., n$. The subspaces can be described as

$$\mathscr{S}_i = \{\boldsymbol{x} \in \mathbb{R}^D : \boldsymbol{x} = \boldsymbol{\mu}_i + \mathbf{U}_i \boldsymbol{y}\},\ i = 1, \cdots, n \qquad (1)$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^D$ is a certain point that can be chosen as $\boldsymbol{\mu}_i = 0$ for linear subspaces while $\boldsymbol{\mu}_i \neq 0$ for affine subspaces. $\mathbf{U}_i$ is a basis for subspace $\mathscr{S}_i$, and $\boldsymbol{y} \in \mathbb{R}_i^d$ is a low-dimensional representation for point $\boldsymbol{x}$ [22]. When the number of subspaces is equal to 1, this problem comes into the widely-applied PCA [23]. The ultimate goal of subspace clustering is to find parameters $n, \{d_i\}_{i=1}^n, \{\boldsymbol{U}_i\}_{i=1}^n$, and $\{\boldsymbol{\mu}_i\}_{i=1}^n$ described above as well as the segmentation of data points with respect to subspaces [22].

Existing studies of subspace clustering can be divided into four main categories: algebraic, iterative, statistical and spectral clustering methods. Algebraic methods, such as matrix factorization-based algorithms [24] [25] [26] and generalized principal component analysis (GPCA) [2] [12], are all designed for linear subspaces. Matrix factorization-based algorithms initialize the segmentation by casting a threshold to the entries of a similarity matrix constructed from the matrix factorization of data points and segment the data with a low-rank factorization of the data matrix and, thus, are a natural extension of PCA from one to multiple independent linear subspaces. However, these methods are theoretically correct under the assumption of independent subspaces, but would fail when the assumption is invalid. Meanwhile, these techniques are not robust to noise. Another problem is that, with less exceptions, they do not compute the number of subspaces and their corresponding dimensions. GPCA fits the data points with a set of polynomials of degree, say m, and the derivatives of which, at a point, give a vector normal to the subspace containing that point. In this case, subspace clustering is the same as fitting and differentiating polynomials. GPCA is computationally cheap when the number of subspaces and their dimensions are small, and meanwhile interceptions between subspaces are naturally allowed in this algorithm. Nonetheless, GPCA is under the assumption of linear subspaces and its computational complexity exponentially expands with the number of subspaces and their dimensions. Plus, the performance of GPCA deteriorates as data become noisy and the nature of this algorithm makes it fairly sensitive to outliers.

Iterative refinement is a relatively efficient technique in improving the performance of algebraic methods on noisy data. An intuitive rundown of each loop is that after an initial segmentation, classical PCA can be used to seek a subspace for each group and then each data point in the dataset can be assigned to its nearest subspace. The iteration of these two procedures can acquire a better approximation of subspaces and segmentation than algebraic algorithms. The aforementioned brief is also the essential idea behind the K-planes algorithm [27], which extends the K-means algorithm [28] from data distribution around numerous cluster centers to hyperplanes.

The K-subspaces algorithm [29] [30] further extends the K-planes algorithm from a collection of hyperplanes to affine subspaces in any dimension. K-subspaces stands out with its simplicity and ability to converge to its local optimum in finite number of iterations. However, the downsides are that it suffers from a rigid requirement on initialization and severe sensitivity to outliers.

The aforesaid two categories seek clustering results using algebraic or geometric properties of data, which makes no exact assumption of distributions of noise or data inside subspaces. Correspondingly, their approximation is unsurprisingly sub-optimal. This issue can be addressed by specifying an appropriate generative model for data, as organized by statistical methods. Prominent statistical approaches include mixture of probabilistic PCA (MPPCA) [31], agglomeraive lossy compression (ALC) [32], etc. Mixture of probabilistic PCA (MPPCA) can be seen as a probabilistic version of K-subspaces. It is simple and intuitive and can be applied to both linear and affine subspaces. However, a crucial drawback of MPPCA is that we need to acquire the number of subpsaces and their corresponding dimensions beforehand. Unlike MPPCA, which targets to obtain a maximum likelihood estimate (MLE) of the parameters inside the mixture model, agglomerative Lossy Compression (ALC) uses a degenerated Gaussian to fit subspaces and searches for the data segmentation that is capable of minimizing the coding length needed to fit these points with a mixture of Gaussian. ALC is able to deal with noisy data points and robust to outliers. Nevertheless, theoretical proof is unavailable for the optimality to this method.

Spectral clustering [33] [34] is a highly desirable technique for clustering high-dimensional datasets. This algorithm first finds an affinity matrix that describes the local information around each point to build an affinity matrix. The affinity measurements might vary in different cases. Then, given a specific affinity matrix, the data segmentation is obtained by the K-means algorithm to the top n eigenvectors of a matrix derived from the affinity matrix, in which n is the number of subspaces (see details in [33]). However, one of the main challenges in applying spectral clustering to subspace clustering problems is to define a decent affinity matrix. Since two data points with a close physical distance do not guarantee to lie in the same subspace (take the points near intersections as an instance) and vice versa. As a consequence, typical distance-based affinity measurements are no longer appropriate in this case.

To obtain better clustering outcomes for subspace clustering, folks have done plenty of trials in constructing an appropriate pairwise affinity matrix for points lying in multiple subspaces. The factorization [24] and GPCA -based affinity [2] [12] are designed only for linear subspaces and still do not well overcome the typical challenge from distance-based affinity matrices in the original spectral clustering algorithm. The local subspace affinity (LSA) [35] and spectral local best-fit flats (SLBF) [36] are established on the observation that a data point and its nearest neighbors (NNs) commonly inhere in an identical subspace. As a result, they fit an affine subspace to each data point and its K-NNs by e.g. PCA, and then define the affinity between two points with the fitted affine subspace. These two methods are more prone to reject outliers and their computational cost is relatively low as well. Nonetheless, LSA has two main limitations. First, it still does not well handle the data points that near intersections since a point's neighbor could belong to a different subspace. In the meantime, the selected neighbors may not span the underlying subspace and thus the designated number for neighbors need to be either small enough so that only points in the same subspace are chosen or large enough so that the neighbors span the local subspace. But the intrinsic nature of SLBF automatically resolves this issue. The affinity construction of locally linear manifold clustering (LLMC) [37] is also based on the thought of fitting a local subspace to a specific point and its K-NNs. Compared to LSA and SLBF, LLMC is more robust to outliers and this approach is also applicable to nonlinear subspaces. However, LLMC suffers from the same two downsides as LSA. What's more, its performance dramatically glides down when it comes to dependent subspaces.

Lately, approaches based on spectral clustering which assume data is self-expressive [38] and drawn from a union of subspaces to form the affinity matrix, which is obtained by solving an optimization problem that incorporates l_1, l_2 or nuclear norm regularization, have become exceedingly trendy. The self-expressiveness property states that each data point can be represented by a linear combination of other points in the union of subspaces. That is,

$$\mathbf{X} = \mathbf{X}\mathbf{C}\ ,\ \mathrm{diag}(\mathbf{C}) = 0 \tag{2}$$

where $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N] \in \mathbb{R}^{D\times N}$ is the data matrix and $\mathbf{C} = [\boldsymbol{c}_1, ..., \boldsymbol{c}_N] \in \mathbb{R}^{N\times N}$ is the matrix of coefficients.

The solution of $\mathbf{C}$ is called subspace preserving since they preserve the clustering pattern of subspaces [21]. Given $\mathbf{C}$, one can build a matrix $\mathbf{A}$ as $\mathbf{A} = |\mathbf{C}| + |\mathbf{C}|^T$, and then apply spectral clustering on $\mathbf{A}$ to cluster data. In the core, all self-expressive subspace clustering approaches manage to handle variants of the optimization problem, which is

$$\min_{\mathbf{C}} ||\mathbf{C}||\ \ s.t.\ \ ||\mathbf{X} - \mathbf{X}\mathbf{C}|| \le \xi,\ \ \mathrm{diag}(\mathbf{C}) = 0. \tag{3}$$

For example, in order to establish an affinity matrix, the sparse subspace clustering (SSC) algorithm [38] [39] proceeds by implementing the basis pursuit (BP) algorithm, which solves the convex l_1 norm optimization problem, so as to reconstruct a sparse representation of data points. Orthogonal matching pursuit (OMP), i.e., a greedy method for sparse model recovery, was developed in [40] [41]. Least squares regression (LSR) [42] employs l_2 regularization to find $\mathbf{C}$. Nuclear norm minimization is applied to low rank subspace clustering (LRSC) [43] to get $\mathbf{C}$. [44] pursues the block-diagonal structure of $\mathbf{C}$ by deriving a graph Laplacian constraint based formulation and then proposes a stochastic subgradient algorithm for optimization. Gao et al. [45] proposes multi-view subspace clustering which conducts clustering on the subspace representation of every view simultaneously. In [46], low-rank representation learning and data segmentation are jointly processed by searching for individual low-rank segmentation as well as implementing the Schatten p-norm relaxation of the non-convex rank objective function. Lastly, [47] obtains the similarity matrix by thresholding the correlations between data points.

Performance-wise of self-expressive-based subspace clustering regimes, the author of [38], [39] shows that while the subspaces are in a disjoint pattern, BP-based methods are more inclined to be subspace preserving. [48], [49], from geometric way of thinking, further analyze BP-based SSC in the cases of intersecting subspaces and data with outliers. These outcomes are expanded to the OMP-based SSC [40], [41] and matching pursuit-based SSC [50] as well. Lately, further generalizations of SSC and Low rank representation (LRR) schemes are presented. In specific, [49] suggests a SSC-based method that jointly conducts clustering and representation learning. [51]–[55] generalize the SSC to handle data with missing information. [56] proposes the temporal subspace clustering algorithm which samples a single data point at each time step and aims at assembling data points into sequential segments, and then come after by clustering the segments into corresponding their subspaces. Nevertheless, none of these methods explores the evolutionary structure of the feature space, which is also the topic discussed in this paper. Previous works are under the assumption that data is obtained in an offline mode and will be fixed without any evolution through time once acquired. Thus, past researches on subspace clustering mostly keep sticking with the so-called static subspace clustering without taking consideration of the inherent evolution structure in datasets.

C. Evolutionary Clustering

Evolutionary clustering, also known as dynamic clustering, is a relatively recent topic formulated in 2006. However, the discussion of evolutionary clustering has been trending in the past few years [57]–[59], and related techniques have been widely applied in a range of practical settings, including prediction of links between blogs [57], identifying communities of spammers [60], tracing parameters of multiantenna communication systems [61], tracking of dynamic networks [61], [62], study in climate change [63], and oceanography [63].

Under the assumption that immediate changes of clustering in a short period of time are not desirable, evolutionary clustering is the problem of organizing timestamped data to generate a clustering sequence by introducing a temporal smoothness framework. Every clustering outcome in the sequence is supposed to be alike to its previous one and meanwhile be able to reflect the data pattern in its current time step. A specific example for this setting is that new data is obtained in daily frequency and will be integrated into the clustering regime everyday. The clustering result should be similar to that from yesterday if the data does not drift from its corresponding historical expectation. Conversely, the clustering consequence need to be modified to reflect the current pattern if data changes dramatically. Therefore, evolutionary clustering algorithms have to trade-off between snapshot quality and temporal cost. Snapshot quality is that the clustering should reflect as precisely as possible the data from the current time step, while temporal cost is that each clustering is not expected to shift dramatically from one to the next time step. A high-quality evolutionary clustering algorithm should be capable of well fitting the data points at each time step as well as generating a smooth cluster evolution that can provide the data analyst with a coherent and easily explainable model.

Chakrabarti et al. [57] originally proposes a generic framework for evolutionary clustering by adding a temporal smoothness penalty term to a static clustering objective function, and this general framework explores K-means and agglomerative hierarchical clustering as illustrative examples. Chi et al. [64], [65] further expands on this idea by suggesting an evolutionary spectral clustering approach. It constructs the following loss function (4) that contains both consistency and smoothness terms

$$\mathbf{C} = \alpha \mathbf{C}_{\text{temporal}} + (1-\alpha)\mathbf{C}_{\text{snapshot}} \tag{4}$$

where $\mathbf{C}_{\text{snapshot}}$ symbolizes the term of consistency, i.e. the current spectral clustering loss, $\mathbf{C}_{\text{temporal}}$ symbolizes that of smoothness, and α is a smoothing parameter to decide the weight of current spectral clustering loss.

Consistency term is generated by spectral clustering which aims at maximizing $\text{tr}(\mathbf{X}^T\mathbf{W}\mathbf{X})$ regarding the graph embedding variables $\mathbf{X}$ and graph similarity matrix $\mathbf{W}$. Equation (4) presumes a certain degree of temporal smoothness between $\mathbf{X}_{t-1}$ and $\mathbf{X}_t$, where the smoothness has the capability of preserving either the Cluster Quality (PCQ) or Cluster Membership (PCM) [64], [65].

In PCQ, $\mathbf{C}_{\text{temporal}}$ is the loss of clustering result at time step t to the similarity matrix at time step $t-1$, which means it will penalize present clustering outcomes that deviate from past similarity results. Whereas in PCM, $\mathbf{C}_{\text{temporal}}$ is established by a measure of distance between clustering result at time step t to the similarity matrix at time step $t-1$. That is to say, it will penalize present clustering outcomes that deviate from past clustering results.

Xu et al. [66] extend the evolutionary spectral clustering algorithm by proposing an adaptive forgetting factor, so as to enable this approach evolve better in the underlying network. This type of decay factor is fairly typical in general dynamic evolutionary network studies in various fields. Ning et al. [67] further apply such models to the domain of blog analysis and show their strong suitability due to the continual updates on the layout of graph and the comparatively efficient evolution that may come up in reaction to news or event of interest. See other evolutionary clustering algorithms that attempt to optimize the cost function (4) in [68]–[71], where the definitions of consistency and smoothness terms vary by approaches.

Rosswog et al. [72] proposes evolutionary extensions of K-means as well as agglomerative hierarchical clustering by filtering the feature vectors using a finite impulse response (FIR) filter which aggregates the estimations of feature vectors. The affinity matrix is then computed among the filter outcomes instead of the feature vectors. This technique fundamentally conducts timely smoothing by expanding the correspondence between points and cluster centers so as to bring in historical information. Nevertheless, it neither can take in varying numbers of clusters over time nor can tackle with items coming and leaving at various time steps.

Also based on the idea of adjusting similarities followed by static clustering, AFFECT algorithm, as an extension to static clustering, is proposed in [73], where the similarity matrix at a specific time t is assumed as the sum of a deterministic matrix, i.e. the affinity matrix, and a Gaussian noise matrix. Nonetheless, for the sake of searching for an optimal smoothing factor α_t, AFFECT makes rather strong assumptions on the structure of affinity matrices , which is generated by assuming a block structure for affinity matrix that only stands when the data at each time step t is a realization of a dynamic Gaussian mixture model. But this is generally not true in practice.

D. Evolutionary Subspace Clustering

For the sake of conducting evolutionary clustering at multiple time steps to temporal data identified by a union-of-subspaces structure, folks probably will think about chaining snapshots from all the time steps together and applying one specific subspace clustering technique w.r.t. this set [21]. Nevertheless, the concatenation will induce a dramatic rise in feature numbers and, thus, lead to unwanted increases in computational complexity. Moreover, by fitting the set with a single union of subspaces, it's almost infeasible to uncover the slight evolutionary changes in the temporal structure of data, which results in an inefficient temporal evolution of subspaces.

We will consider the real-time motion segmentation task [74], [75] as an illustration over here, where the aim of this task is to recognize and track motions in a video sequence. The problem of real-time motion segmentation is associated with that of offline motion segmentation [38], [39]. The difference between them is that by taking use of all the frames in the sequence, clustering is performed only once in the offline scenario, whereas by receiving each snapshot of the sequence one at a time, clustering is performed step by step in the online setting. The generated subspaces depict the evolution of motions, in which subspaces in closer snapshots are similar while in more far-apart ones may significantly deviate. Thus, as in the aforesaid C&C approach, imposing a sole subspace layout for the whole time sequence may incur poor clustering outcomes. And a regime that is capable of judiciously taking advantage of the evolutionary structure while preserving the union-of-subspaces structure is in need.

Instead of assuming a common subspace for all clusters, Vahdat et al. [76]–[78] proposes a bottom-up symbiotic evolutionary subspace clustering (S-ESC) algorithm consisting of four steps. In specific, it first proceeds by a regular static clustering algorithm to attribute-wisely establish the lattice from which subspace clusters will be designed. The second step co-evolves cluster centroids (symbionts) as well as clustering solutions (hosts) alongside one another through a multi-objective framework. Thirdly, fitness evaluation is performed relative to host individuals (i.e., clustering solutions). And the last step is to to conduct a search throughout different cluster combinations so as to find the best combination of subspace clusters.

In [59], [65], [72], [73], authors propose evolutionary subspace clustering techniques by employing static clustering algorithms such as spectral clustering, etc. to process the affinity matrix and then apply equation (5) for evolutionary smoothing. The specific procedures might vary in each case.

$$\mathbf{C}_t = \alpha_t \bar{\mathbf{C}}_t + (1 - \alpha_t)\mathbf{C}_{t-1} \tag{5}$$

In equation (5), $\bar{\mathbf{C}}_t$ and $\mathbf{C}_{t-1}$ denote the affinity matrix constructed solely from $\mathbf{X}_t$ and smoothed affinity matrix at time $t-1$ respectively, and α_t is the smoothing parameter at time t. The affinity matrix $\bar{\mathbf{C}}_t$ is constructed from $\mathbf{X}_t$ using general similarity notions such as the negative Euclidean distance of the data points or its exponential variant. Nonetheless, while $\bar{\mathbf{C}}_t$ is learnt from (5), it does not consider the representation of data points in previous time steps, which will lead to suboptimal performance in subspace clustering applications. Plus, except AFFECT algorithm [73], none of the aforementioned model provides a procedure for finding the smoothing parameter α_t.

Lately, to get the above mentioned issues solved, Hashemi et al. [21] propose an ESCM framework that exploits the self-expressiveness property of data to learn a representation for $\mathbf{X}_t$ and meanwhile takes into account of data representation learnt in the previous time step. Moreover, they rely on alternating minimization to automatically learn the smoothing parameter α_t at each time step. The general idea of ESCM framework is listed in (6)-(7).

$$\mathbf{C}_t = f_\theta(\mathbf{C}_{t-1}), \quad \mathbf{X}_t = \mathbf{X}_t\mathbf{C}_t, \quad \operatorname{diag}(\mathbf{C}_t) = \mathbf{0}, \tag{6}$$

where

$$f_\theta(\mathbf{C}_{t-1}) \in \mathscr{P}_C$$

The subspace clustering representation $\mathbf{C}_t$ is reckoned as a matrix-valued function solved by θ, which exploits the self-expressive characteristic inherited in data and meanwhile, in a certain way, links $\mathbf{C}_t$ to the preceding representation matrix $\mathbf{C}_{t-1}$. In theory, the function $f_\theta : \mathscr{P}_C \to \mathscr{P}_C$ could be any parametric function while the set $\mathscr{P}_C \subseteq \mathbb{R}^{N_t \times N_t}$ stands for any preferred parsimonious structure imposed on the representation matrices at each time instant, e.g., sparse or low-rank representations.

In the scenario of [21], $f_\theta(\mathbf{C}_{t-1})$ is defined as equation (7). The values of parameters $\theta = (\mathbf{U}, \alpha)$ specify the relationship between $\mathbf{C}_{t-1}$ and $\mathbf{C}_t$, and need to be learned from data. Intuitively, the innovation representation matrix $\mathbf{U}$ captures changes in the representation of data points between consecutive time steps. The other term on the right-hand side of (7), $(1-\alpha)\mathbf{C}_{t-1}$, is the part of temporal representation that carries over from the previous snapshot of data.

$$\mathbf{C}_t = f_\theta(\mathbf{C}_{t-1}) = \alpha\mathbf{U} + (1-\alpha)\mathbf{C}_{t-1}. \tag{7}$$

Though [21] has done a successful trial of evolutionary subspace clustering with its proposed ESCM framework, merely employing (7) on time-series smoothing makes the whole model lack of expressiveness and unable to efficiently unveil the substantial evolutionary information behind data. In this paper, to address the aforementioned issue, we propose to substitute (7) with recurrent LSTM deep networks for the reason of significant performance of LSTM in learning complicate temporal patterns under various settings. As the experimental results demonstrate, our proposed framework does effectively capture the temporal behaviors of data and achieve a significantly better accuracy than previous ones, which further prove the fitting and capability of LSTM networks in processing temporal information.

III. LSTM Evolutionary Subspace Clustering

In this section, we will present our model for the evolutionary subspace clustering based on the recurrent neural networks LSTM.

Let $\{\mathbf{x}_{t,i}\}_{i=1}^{N_t}$ be a set of real-valued D_t dimensional vectors at time t, and we aggregate them into a matrix as $\mathbf{X}_t = [\mathbf{x}_{t,1}, ..., \mathbf{x}_{t,N_t}] \in \mathbb{R}^{D_t \times N_t}$. All the data points at time step t are drawn from a union of n_t evolving subspaces $\{\mathscr{S}_{t,i}\}_{i=1}^{n_t}$ with dimensions $\{d_{t,i}\}_{i=1}^{n_t}$. And the full dataset is $\mathscr{D} = \{\mathbf{X}_t\}_{t=1}^{T}$. Without a loss of generality, we assume that the columns of $\mathbf{X}_t$ are normalized vectors with unit l_2 norm. Due to the underlying union of subspaces structure, the data points at each time step themselves satisfy the self-expressiveness property [38] formally stated in section II-B equation (2).

The main purpose of most existing static subspace clustering algorithms is to partition $\{\mathbf{x}_{t,i}\}_{i=1}^{N_t}$ into n_t groups so as to allocate data points that belong to the same subspace into the same cluster. Manipulating the self-expressive characteristic of data does improve the progress of exploiting the fact that a collection of data points belongs to a union of subspaces.

In numerous applications of subspace clustering, apart from lying in a union of subspaces, data is also of temporal patterns, which makes researchers get started on exploring the self-expressiveness based evolutionary subspace clustering. Still, the past studies focus on applying weighted average akin methods or other relatively simple temporal processing models to smooth the temporal clustering results at multiple time steps [21], with less people taking advantage of the capability of deep learning methods in learning data evolution. Upon their chain-like nature, recurrent neural networks (RNNs) have been showed to have excellent performance on persisting past information, which naturally works for scenarios of processing various types of sequences. Further, Long short term memory (LSTM), a very special kind of RNNs well capable of learning long term dependencies, works much better than the standard version RNNs for many tasks [79]. To this end, we propose to find a representation matrix $\mathbf{C}_t$ for each time t using LSTM networks, such that f_θ in equation (6) is instead implemented by LSTM networks. We will refer to our proposed evolutionary subspace clustering scheme that satisfies equation (6), with f_θ expressed by LSTM networks, as LSTM evolutionary subspace clustering method (LSTM-ESCM) in what follows.

LSTM networks is of three inputs - cell and hidden states generated from time step $t-1$ and data information at t, two outputs - processed cell and hidden states for timestamp t, and in between are multiple gates to optionally let information through. In our scenario, as shown in **Fig. 1**, we use $\mathbf{M}_{t-1}$ and $\mathbf{M}_t$ to denote the cell states at time step $t-1$ as well as t, which represent the memories of previous and current blocks. $\mathbf{C}_{t-1}$ and $\mathbf{C}_t$ are hidden states, i.e. our smoothed clustering results, at time step $t-1$ and t. $\mathbf{X}_t$ is our input data at time point t. And then the proposed LSTM-ESCM is proceeded by equations in (8), where $\widetilde{\mathbf{M}}_t$, f_t, i_t, and o_t are new cell state candidate values, and forget, input, and output gates of current block at timestamp t correspondingly.

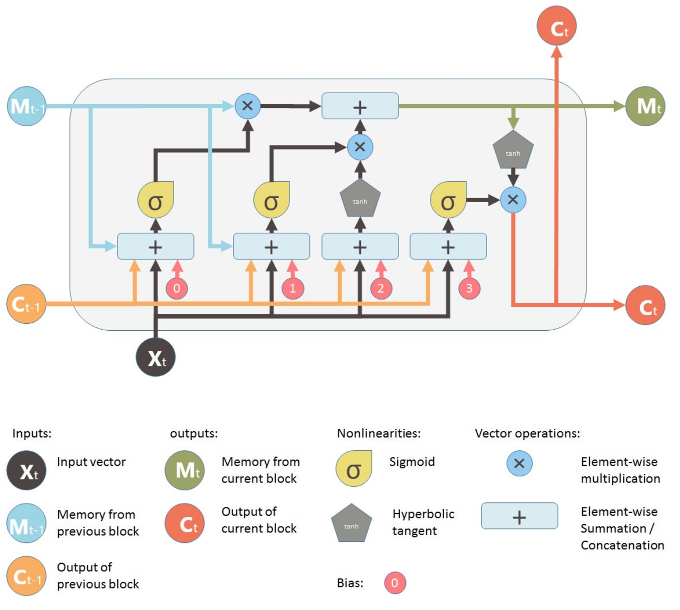

Figure 1. LSTM network structure of LSTM-ESCM

$$\begin{aligned} f_t &= \sigma(W_f \cdot [\mathbf{C}_{t-1}, \mathbf{X}_t] + b_f) \\ i_t &= \sigma(W_i \cdot [\mathbf{C}_{t-1}, \mathbf{X}_t] + b_i) \\ \widetilde{\mathbf{M}}_t &= \tanh(W_m \cdot [\mathbf{C}_{t-1}, \mathbf{X}_t] + b_m) \\ \mathbf{M}_t &= f_t * \mathbf{M}_{t-1} + i_t * \widetilde{\mathbf{M}}_t \\ o_t &= \sigma(W_o \cdot [\mathbf{C}_{t-1}, \mathbf{X}_t] + b_o) \\ \mathbf{C}_t &= o_t * \tanh(\mathbf{M}_t) \end{aligned} \tag{8}$$

For the sake of obtaining a such representation matrix $\mathbf{C}_t$, we construct the following optimization regime

$$\begin{aligned} &\min_\theta \frac{1}{2}\|\mathbf{X}_t - \mathbf{X}_t f_\theta(\mathbf{C}_{t-1})\|_F^2 \\ &s.t. \;\; f_\theta(\mathbf{C}_{t-1}) \in \mathscr{P}_C, \quad \mathrm{diag}(\mathbf{C}_t) = \mathbf{0} \end{aligned} \tag{9}$$

and employ the derived representation matrix $\mathbf{C}_t = f_\theta(\mathbf{C}_{t-1})$ to conduct data segmentation. After getting a solution to (9) by solving the general constrained representation learning problem, we build an affinity matrix $\mathbf{A}_t = |\mathbf{C}_t| + |\mathbf{C}_t|^T$ and then apply spectral clustering to $\mathbf{A}_t$.

IV. Implementation of LSTM-ESCM

MATLAB deep learning toolbox is utilized to implement the proposed LSTM-ESCM framework. See [80] in details. It constructs networks by connecting layers. Since the toolbox has yet strong requirements on data structures, we hybrid a couple of predefined layers implanted in the toolbox - sequence input layer [81], LSTM layer [82], and fully connected layer [83] - and our customized ones - padding layer and regression layer - so as to realize the LSTM-ESCM schemes. Specific implementation is discussed in the following subsections.

A. Loss Function

MATLAB deep learning toolbox uses a fairly tight data structure for its layer definition, which means there is not much freedom to define an appropriate loss function, in our case. The strong requirement that the size and dimension of the last layer output must be the same as those of the target obstacles us to implement our loss function. Here, we will take the loss at one timestamp t as an illustration.

$$L = \frac{1}{2}\|\mathbf{X}_t - \mathbf{X}_t\mathbf{C}_t\|_F^2.$$

Obviously, as shown in **Fig. 1**, the output of our last layer is the clustering result $\mathbf{C}_t \in \mathbb{R}^{N_t\times N_t}$ while the target is $\mathbf{X}_t \in \mathbb{R}^{D_t\times N_t}$, where the dimensions of those two do not align. But fortunately we can rewrite the loss as

$$\begin{aligned} L = \frac{1}{2}\|\mathbf{X}_t - \mathbf{X}\mathbf{C}_t\|_F^2 &= \frac{1}{2}\|\mathbf{X}_t(\mathbf{I} - \mathbf{C}_t)\|_F^2 \\ &= \frac{1}{2}\text{tr}((\mathbf{I} - \mathbf{C}_t)^T\mathbf{X}_t^T\mathbf{X}_t(\mathbf{I} - \mathbf{C}_t)) \\ &= \frac{1}{2}\text{tr}((\mathbf{X}_t^T\mathbf{X}_t)((\mathbf{I} - \mathbf{C}_t)(\mathbf{I} - \mathbf{C}_t^T))) \end{aligned} \tag{10}$$

and its corresponding derivative is

$$\frac{\partial L}{\partial \mathbf{C}_t} = (\mathbf{X}_t^T\mathbf{X}_t)(\mathbf{C}_t - \mathbf{I})$$

To penalize out sparsity for $\mathbf{C}_t$, we may add the following regularizer to the loss function

$$R = \lambda\|\mathbf{C}_t\|_1 = \lambda\sum_{ij}|c_{ij}| \tag{11}$$

and

$$\frac{\partial R}{\partial \mathbf{C}_t} = \lambda\text{sign}(\mathbf{C}_t).$$

Thus, we can finalize our loss function and its derivatives as

$$L_\lambda = \frac{1}{2}\text{tr}((\mathbf{X}_t^T\mathbf{X}_t)((\mathbf{I} - \mathbf{C}_t)(\mathbf{I} - \mathbf{C}_t^T))) + \lambda\|\mathbf{C}_t\|_1 \tag{12}$$

$$\frac{\partial L_\lambda}{\partial \mathbf{C}_t} = (\mathbf{X}_t^T\mathbf{X}_t)(\mathbf{C}_t - \mathbf{I}) + \lambda\text{sign}(\mathbf{C}_t) \tag{13}$$

We know the size of $\mathbf{X}_t^T\mathbf{X}_t \in \mathbb{R}^{N_t\times N_t}$, as new targets, is same as the size of $\mathbf{C}_t \in \mathbb{R}^{N_t\times N_t}$, as the output from the network.

B. Data Structure

Another issue with MATLAB deep learning toolbox is that the LSTM layer only accepts vector sequence, thus we have to convert our inputs and targets into vectors.

At any time point t, we have a data matrix $\mathbf{X}_t$ containing N_t key points (to be clustered) in columns while the rows corresponding to D_t features of objects. That is $\mathbf{X}_t \in \mathbb{R}^{D_t\times N_t}$. We have to assume we always have N_t key points at any time step. How to deal with different number of key points at different time step is another research topic to be discussed in our future work.

For a full dataset $\mathscr{D}$, it is a general practice to break down the time series into shorter sections of sequence. For example, using a moving window to slide out a length S_t shorter sequence, or cutting the entire sequence into many shorter sequences without overlapping. The number of such shorter sequences is denoted by N, i.e., the training number (or the total number of training and testing), and these sequences will be sent into networks for training purpose individually. Our data is processed by the former option.

Under MATLAB requirement, we need to organize each “data” (a shorter sequence) in the following matrix

$$\mathbf{X}^i = [\text{vec}(\mathbf{X}_{t_i+1}), ..., \text{vec}(\mathbf{X}_{t_i+s})] \in \mathbb{R}^{(D_t\times N_t)\times S_t}$$

where $i = 1, 2, ..., N$. All these training data in matrix should be organized into MATLAB cells of size $N \times 1$, that is,

$$\mathscr{X}_{\text{train}} = \{\mathbf{X}^1, ..., \mathbf{X}^N\}$$

Given the special loss function defined in (12), we have to organize the target data as

$$\mathbf{Y}^i = [\text{vec}(\mathbf{X}_{t_i+1}^T\mathbf{X}_{t_i+1}), ..., \text{vec}(\mathbf{X}_{t_i+s}^T\mathbf{X}_{t_i+s})] \in \mathbb{R}^{N_t^2\times S_t}$$

and the target data $\mathscr{Y}_{\text{train}}$ will be organized into MATLAB cells of size $N \times 1$ as well, which is

$$\mathscr{Y}_{\text{train}} = \{\mathbf{Y}^1, ..., \mathbf{Y}^N\}$$

Same, the output clustering and affinity results for each sequence will be

$$\mathbf{C}^i = [\text{vec}(\mathbf{C}_{t_i+1}), ..., \text{vec}(\mathbf{C}_{t_i+s})] \in \mathbb{R}^{N_t^2\times S_t}$$

$$\mathbf{A}^i = [\text{vec}(\mathbf{A}_{t_i+1}), ..., \text{vec}(\mathbf{A}_{t_i+s})] \in \mathbb{R}^{N_t^2\times S_t}$$

and all these outputs will be stored into MATLAB cells of size $N \times 1$ as

$$\mathscr{C} = \{\mathbf{C}^1, ..., \mathbf{C}^N\}$$

$$\mathscr{A} = \{\mathbf{A}^1, ..., \mathbf{A}^N\}$$

C. Network Architecture

We will construct a LSTM architecture accepting the sequences of $\mathbf{X}^i$ as input for training purpose. The training process of each sequence is completely independent. Meanwhile, during training, only a single column of $\mathbf{X}^i$ will be sent into the network architecture at each time step. Hence, in each t, the data dimension is $D_tN_t \times 1$. As each target in the target sequence is in dimension $N_tN_t \times 1$, directly taking the output from the LSTM to the target will need a hidden dimension of $N_tN_t \times 1$. This will produce huge numbers of weight parameters in LSTM at scale of $O(D_t \times N_t^3)$. We will use an LSTM of a much smaller hidden input size h and then use a fully connected layer to connect LSTM with the target.

As constrained in (6), we do not expect the network to produce the diagonal elements of each $\mathbf{C}_t \in \mathbb{R}^{N_t \times N_t}$. Therefore, the number of entries of the output from the fully connected layer should be $N_t^2 - N_t$. To match Matlab requirement on the Regression Layer, we will custom a padding layer to add zeros back to the diagonal of $\mathbf{C}_t$ to make sure $\text{diag}(\mathbf{C}_t) = 0$. To this end, we have defined two custom layers: one for padding and one for loss. Plus, the new custom regression layer will take a parameter for $\lambda > 0$.

The network can be defined in MATLAB code as

```
layers = [ ...
  sequenceInputLayer(D_t*N_t)
  lstmLayer(h, 'OutputMode', 'sequence')
  fullyConnectedLayer(N_t*N_t-N_t)
  myPaddingLayer(N_t)
  myRegressionLayer('Evolving', λ)];
```

Finally, the overall proposed LSTM-ESCM framwork can be summarized as **Algorithm 1**.

Algorithm 1: The Proposed LSTM-ESCM Algorithm

```
Input: Training dataset $\mathscr{X}_{\text{train}}$. Target dataset $\mathscr{Y}_{\text{train}}$.
    Regulariser $\lambda$. Hidden size $h$. Learning rate $\eta$.
1: for $i = 1, 2, ..., N$ do
2:     Apply (6), (8), and (9) to conduct clustering training
       with LSTM networks and generate $\mathbf{C}^i$;
3:     Construct affinity matrix $\mathbf{A}^i$ from $\mathbf{C}^i$;
4:     Implement spectral clustering to $\mathbf{A}^i$ to obtain data
       segmentation;
5: end for
```

V. Experiment on Real Time Motion Segmentation

The motion segmentation problem - recovering scene geometry and camera motion from a sequence of images - is a very crucial pre-processing step for multiple applications in computer vision, such as surveillance, tracking, action recognition, etc, and has attracted much of the attention of the vision community over the last decade [12], [84]. The problem is formed as clustering a set of two dimensional trajectories extracted from a video sequence with several rigidly moving objects into groups; the resulting clusters represent different spatial-temporal regions.

The video sequence is often times obtained as a stream of frames and it is mostly processed in a real-time mode [74], [75]. In the real-time setting, the tth snapshot of $\mathbf{X}_t$ (a time interval consisting of multiple video frames) is of dimension $2F_t \times N_t$, where N_t is the number of trajectories at tth time interval, F_t is the number of video frames received in tth time interval, n_t is the number of rigid motions at tth time interval, and $F = \sum_t F_t$ denotes the total number of frames [21].

As the obtained video sequence is identified by its temporal pattern, the real-time motion segmentation problem has a good chance of getting solved by evolutionary subspace clustering algorithms. Specifically, the trajectories of n_t rigid motions sit in a set of n_t low-dimensional subspaces in $\mathbb{R}^{2F_t}$ at tth snapshot, each having no more than $d_t = 3n_t$ dimension [84]. In contrast, offline evolutionary subspace clustering is conducted on the whole video sequence, and thus we will expect much higher accuracy than that in the online structures. Whereas, offline modes solely limit in several specific situations and cannot be well extended to the settings where a few motions gradually vanish or new motions come up in the video sequence [21]. To validate the performance of the proposed LSTM-ESCM framework, Hopkins 155 database [12] is considered here.

Table I
Distribution of the number of points and frames.

	2 Groups			3 Groups		
	# Seq.	Points	Frames	# Seq.	Points	Frames
Check.	78	291	28	26	437	28
Traffic	31	241	30	7	332	31
Articul.	11	155	40	2	122	31
All	120	266	30	35	398	29
Point Distr.	35-65			20-24-56		

A. Hopkins 155 Dataset

The database collects 50 video sequences of indoor and outdoors scenes containing two or three motions, in which each video sequence $\mathbf{X}$ with three motions was split into three motion sequences $\mathbf{X}_{\text{g12}}$, $\mathbf{X}_{\text{g13}}$ and $\mathbf{X}_{\text{g23}}$ containing the points from groups one and two, one and three, and two and three, respectively. This gives a total of 155 motion sequences: 120 with two motions and 35 with three motions, in which the number of checkerboard sequences is 104, traffic is 38, and articulated/non-rigid is 13.

Checkerboard sequences are indoor scenes taken with a handheld camera under controlled conditions. The checkerboard pattern on the objects is used to assure a large number of tracked points. Traffic scenes are taken by a moving handheld camera and most of them contain degenerate motions, particularly linear and planar motions. Articulated/non-rigid sequences display motions constrained by joints, head and face motions, people walking, etc [12].

The entire database is available at [85]. **Table I** reports the number of sequences and the average number of tracked points and frames for each category. In the entire dataset, per sequence, the number of points ranges from 39 to 556, while frames from 15 to 100. Point distribution represents the average distribution of points per moving object, where the last group corresponds to the camera motion (motion of the background). The statistic in table is solely computed based on the original 50 videos [12].

B. Experiment

In contrast to most of the previous work which handles the aforedescribed dataset in an offline mode, we will process it in a real-time setting. Each video is divided into T data matrices $\{\mathbf{X}_t\}_{t=1}^T$ so that $F_t \geq 2n$ for a video sequence with n motions. Next, PCA is applied on $\mathbf{X}_t$ and the top $D = 4n$ singular vectors are kept as the final input to the representation learning algorithms. The input parameters for LSTM-ESCM are set as

Table II
PERFORMANCE COMPARISONS OF STATIC SUBSPACE CLUSTERING, AFFECT AND CESM BENCHMARK ALGORITHMS ON HOPKINS 155 DATASET

Learning Method	Static		AFFECT		CESM	
	error (%)	runtime (s)	error (%)	runtime (s)	error (%)	runtime (s)
BP	10.76	1.92	9.86	1.29	8.77	1.32
OMP	31.66	0.06	14.47	0.03	6.85	0.03
AOLS ($L = 1$)	16.27	0.27	9.27	0.20	8.24	0.22
AOLS ($L = 2$)	8.54	0.5	6.17	0.23	5.70	0.25
AOLS ($L = 3$)	6.97	0.76	5.92	0.26	5.60	0.28

Table III
PERFORMANCE OF PROPOSED FRAMEWORK ON HOPKINS 155 DATASET

Learning Method	smoothing		Test on 1 snapshot	Test on 2 snapshots
	error (%)	runtime (s)	error (%)	error (%)
Proposed LSTM-ESCM	0.005	0.13	4.42	4.42

$\lambda = 0.1$, $h = \text{ceil}(\frac{N_t}{5})$ and $\eta = 0.001$ for each sequence. Note that N_t is the same for each sequence.

A specific static subspace clustering algorithm [20], AFFECT [73] ,and CESM [21] are used as benchmarks for our proposed framework. Static subspace clustering applies subspace clustering at each time step independently from the previous outcomes; AFFECT and CESM both applies spectral clustering [33] on the weighted average of affinity matrices $\mathbf{A}_t$ and $\mathbf{A}_{t-1}$. The default choices for the affinity matrix in AFFECT are the negative squared Euclidean distance or its exponential form, while the affinity matrices in CESM are explored under the self-expressive properties [38] of $\mathbf{X}$. Under its original settings, AFFECT achieves a clustering error of 44.1542% and 21.9643% using the negative squared Euclidean distance or its exponential form, respectively, which, as presented in **Table II**, is inferior even to the static subspace clustering algorithms. Hence, to fairly compare the performance of different evolutionary clustering strategies, BP [38], [39], [86], OMP [40], [41], [87], and AOLS ($L = 1, 2, 3$) [88], [89] are employed to learn the representations for all the benchmark schemes, including AFFECT. Plus, the results are averaged over all sequences and time intervals excluding the initial time interval $t = 1$. The initial time interval is excluded because for a specific representation learning method (e.g., BP), the results of static as well as evolutionary subspace clustering coincide [21].

The performances of multiple benchmark schemes are presented in **Table II**, where all the error rates are smoothing (training) errors, while the performances of our proposed LSTM-ESCM framework are shown in **Table III**, in which both smoothing errors as well as test errors on the last one or two snapshots of each sequence are displayed. Note that for the CESM algorithm, though authors in [21] list results corresponding to both a constant smoothing factor and a smoothing factor achieved by using their proposed alternating minimization schemes, we will solely keep the later one as our benchmark.

As presented in **Table II**, for all the representation learning schemes, static subspace clustering has the highest clustering errors among all benchmark models due to not incorporating any knowledge about the representations of the data points at other times, while CESM performs relatively better than its AFFECT counterparts. For runtimes, methods rooted on OMP are the most time-efficient, whereas BP-based approaches have the slowest processing speed. The outcomes in **Table III** show that the cluster errors of LSTM-ESCM is fairly promising and the training error is not even in the same scale as those in **Table II** and meanwhile even the test clustering outcomes perform better than the training error rates of all our benchmark models. Plus, apart from a bit slower than methods based on the OMP optimization regime, the runtime of our proposed framework is also quite superior.

VI. CONCLUSION

In this paper, we research on evolutionary subspace clustering, the problem of arranging a set of evolving data points which in actual fact lie in a union of low-dimensional evolving subspaces, and proposed the LSTM-ESCM framework to cope with the related applications. Our proposed model takes advantage of the self-expressive property behind data so as to learn out the parsimonious representation of data at each timestamp while using LSTM deep networks to conduct temporal information learning over the whole data sequence. Due to the limitation of MATLAB deep learning toolbox, we combine its predefined sections as well as our customized functions to realize our proposed model. Then, the experiment is executed on a real-world motion segmentation problem using the well know dataset - JHU motion 155. The experimental outcomes demonstrate that compared to the benchmark models, our proposed one remarkably dominates in the case of both run time and accuracy.

Nonetheless, even after adding in a fully connected layer to lower down the computational cost of LSTM-ESCM, this model is for now still only capable of processing small to medium

-scale datasets. Thus, how to expand our algorithm to solve the computational infeasibility of real-world high-dimensional data is of great interest.

For future research interests, there are four directions to be pointed out. First, it would be worthwhile to further expand the LSTM-ESCM scheme to other subspace clustering methods, such as the approaches relying on seeking low-rank representations, etc. Second, analysis of the theoretical foundation of subspace clustering to interpret and analyze the performance of the proposed model would be meaningful. Next, apart from LSTM networks, other neural network architectures could also be exploited for conducting temporal smoothing. Lastly, how to apply LSTM-ESCM to other evolving learning tasks, such as the topic of variational continual learning [90], is also of great interest to explore.

References

[1] A. Y. Yang, J. Wright, Y. Ma, and S. S. Sastry, "Unsupervised segmentation of natural images via lossy data compression," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 212–225, 2008.

[2] R. Vidal, R. Tron, and R. Hartley, "Multiframe motion segmentation with missing data using powerfactorization and gpca," *International Journal of Computer Vision*, vol. 79, no. 1, pp. 85–105, 2008.

[3] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *CVPR (1)*, 2003, pp. 11–18.

[4] W. Hong, J. Wright, K. Huang, and Y. Ma, "Multiscale hybrid linear models for lossy image representation," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3655–3671, 2006.

[5] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "An algebraic geometric approach to the identification of a class of linear hybrid systems," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 1. IEEE, 2003, pp. 167–172.

[6] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Advances in neural information processing systems*, 2009, pp. 2080–2088.

[7] H. Xu, C. Caramanis, and S. Sanghavi, "Robust pca via outlier pursuit," in *Advances in Neural Information Processing Systems*, 2010, pp. 2496–2504.

[8] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011.

[9] B. Yang, "Projection approximation subspace tracking," *IEEE Transactions on Signal processing*, vol. 43, no. 1, pp. 95–107, 1995.

[10] M. Rahmani and G. K. Atia, "High dimensional low rank plus sparse matrix decomposition," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2004–2019, 2017.

[11] ——, "Randomized robust subspace recovery and outlier detection for high dimensional data matrices," *IEEE Transactions on Signal Processing*, vol. 65, no. 6, pp. 1580–1594, 2017.

[12] R. Tron and R. Vidal, "A benchmark for the comparison of 3-d motion segmentation algorithms," in *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–8.

[13] X. Cao, Y. Zhong, Y. Zhou, J. Wang, C. Zhu, and W. Zhang, "Interactive temporal recurrent convolution network for traffic prediction in data centers," *IEEE Access*, vol. 6, pp. 5276–5289, 2018.

[14] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *International Conference on Machine Learning*, no. 34, 2017, pp. 1–5.

[15] Z. C. Lipton, D. C. Kale, and R. C. Wetzel, "Phenotyping of clinical time series with lstm recurrent neural networks," *arXiv preprint arXiv:1510.07641*, 2015.

[16] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, 2016, pp. 324–328.

[17] M. Wöllmer, F. Eyben, B. Schuller, E. Douglas-Cowie, and R. Cowie, "Data-driven clustering in emotional space for affect recognition using discriminatively trained lstm networks," in *Proc. Interspeech 2009, Brighton, UK*, 2009, pp. 1595–1598.

[18] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[19] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou, "Patient subtyping via time-aware lstm networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2017, pp. 65–74.

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

[21] A. Hashemi and H. Vikalo, "Evolutionary self-expressive models for subspace clustering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 6, pp. 1534–1546, 2018.

[22] R. Vidal, "Subspace clustering," *Signal Processing Magazine IEEE*, vol. 28, pp. 52 – 68, 04 2011.

[23] I. Jolliffe, "Principal component analysis," *Technometrics*, vol. 45, no. 3, p. 276, 2003.

[24] T. E. Boult and L. G. Brown, "Factorization-based segmentation of motions," in *Proceedings of the IEEE workshop on visual motion*. IEEE, 1991, pp. 179–186.

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

[26] C. W. Gear, "Multibody grouping from motion images," *International Journal of Computer Vision*, vol. 29, no. 2, pp. 133–150, 1998.

[27] P. S. Bradley and O. L. Mangasarian, "k-plane clustering," *J. of Global Optimization*, vol. 16, no. 1, pp. 23–32, Jan. 2000. [Online]. Available: https://doi.org/10.1023/A:1008324625522

[28] R. Duda, P. Hart, and D. Stork, "Pattern classification. 2nd edn wiley," *New York*, vol. 153, 2000.

[29] P. Tseng, "Nearest q-flat to m points," *Journal of Optimization Theory and Applications*, vol. 105, no. 1, pp. 249–252, 2000.

[30] P. K. Agarwal and N. H. Mustafa, "K-means projective clustering," in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2004, pp. 155–165.

[31] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[32] H. Derksen, Y. Ma, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy coding and compression," in *Visual Communications and Image Processing 2007*, vol. 6508. International Society for Optics and Photonics, 2007, p. 65080H.

[33] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.

[34] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[35] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *European conference on computer vision*. Springer, 2006, pp. 94–106.

[36] T. Zhang, A. Szlam, Y. Wang, and G. Lerman, "Hybrid linear modeling via local best-fit flats," *International journal of computer vision*, vol. 100, no. 3, pp. 217–240, 2012.

[37] A. Goh and R. Vidal, "Segmenting motions of different types by unsupervised manifold clustering," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–6.

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

[39] ——, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[40] E. L. Dyer, A. C. Sankaranarayanan, and R. G. Baraniuk, "Greedy feature selection for subspace clustering," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2487–2517, 2013.

[41] C. You, D. Robinson, and R. Vidal, "Scalable sparse subspace clustering by orthogonal matching pursuit," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3918–3927.

[42] C. Lu, J. Feng, Z. Lin, and S. Yan, "Correlation adaptive subspace segmentation by trace lasso," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1345–1352.

[43] 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.

[44] J. Feng, Z. Lin, H. Xu, and S. Yan, “Robust subspace segmentation with block-diagonal prior,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3818–3825.

[45] H. Gao, F. Nie, X. Li, and H. Huang, “Multi-view subspace clustering,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4238–4246.

[46] F. Nie and H. Huang, “Subspace clustering via new low-rank model with discrete group structure constraint.” in *IJCAI*, 2016, pp. 1874–1880.

[47] R. Heckel and H. Bölcskei, “Robust subspace clustering via thresholding,” *IEEE Transactions on Information Theory*, vol. 61, no. 11, pp. 6320–6342, Nov 2015.

[48] M. Soltanolkotabi, E. J. Candes *et al.*, “A geometric analysis of subspace clustering with outliers,” *The Annals of Statistics*, vol. 40, no. 4, pp. 2195–2238, 2012.

[49] C.-G. Li, C. You, and R. Vidal, “Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework,” *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2988–3001, 2017.

[50] M. Tschannen and H. Bölcskei, “Noisy subspace clustering via matching pursuits,” *IEEE Transactions on Information Theory*, vol. 64, no. 6, pp. 4081–4104, 2018.

[51] E. Elhamifar, “High-rank matrix completion and clustering under self-expressive models,” in *Advances in Neural Information Processing Systems*, 2016, pp. 73–81.

[52] C.-G. Li and R. Vidal, “A structured sparse plus structured low-rank framework for subspace clustering and completion.” *IEEE Trans. Signal Processing*, vol. 64, no. 24, pp. 6557–6570, 2016.

[53] J. Fan and T. W. Chow, “Sparse subspace clustering for data with missing entries and high-rank matrix completion,” *Neural Networks*, vol. 93, pp. 36–44, 2017.

[54] Z. Charles, A. Jalali, and R. Willett, “Subspace clustering with missing and corrupted data,” *arXiv preprint arXiv:1707.02461*, 2017.

[55] M. C. Tsakiris and R. Vidal, “Theoretical analysis of sparse subspace clustering with missing entries,” *arXiv preprint arXiv:1801.00393*, 2018.

[56] S. Li, K. Li, and Y. Fu, “Temporal subspace clustering for human motion segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4453–4461.

[57] D. Chakrabarti, R. Kumar, and A. Tomkins, “Evolutionary clustering,” in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’06. New York, NY, USA: ACM, 2006, pp. 554–560. [Online]. Available: http://doi.acm.org/10.1145/1150402.1150467

[58] Y. Wang, S.-X. Liu, J. Feng, and L. Zhou, “Mining naturally smooth evolution of clusters from dynamic data,” in *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 125–134.

[59] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, “Evolutionary spectral clustering by incorporating temporal smoothness,” in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’07. New York, NY, USA: ACM, 2007, pp. 153–162. [Online]. Available: http://doi.acm.org/10.1145/1281192.1281212

[60] K. S. Xu, M. Kliger, Y. Chen *et al.*, “Tracking communities of spammers by evolutionary clustering,” in *International Conference on Machine Learning Workshop on Social Analytics: Learning from Human Interactions*, 2010.

[61] N. Czink, R. Tian, S. Wyne, F. Tufvesson, J. Nuutinen, J. Ylitalo, E. Bonek, and A. F. Molisch, “Tracking time-variant cluster parameters in mimo channel measurements,” in *2007 Second International Conference on Communications and Networking in China*, Aug 2007, pp. 1147–1151.

[62] F. Folino and C. Pizzuti, “An evolutionary multiobjective approach for community discovery in dynamic networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1838–1852, Aug 2014.

[63] S. Gnnemann, H. Kremer, C. Laufktter, and T. Seidl, “Tracing evolving subspace clusters in temporal climate data,” *Data Mining and Knowledge Discovery*, vol. 24, no. 2, pp. 387–410, Mar 2012. [Online]. Available: https://doi.org/10.1007/s10618-011-0237-7

[64] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, “Evolutionary spectral clustering by incorporating temporal smoothness,” in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 153–162.

[65] ——, “On evolutionary spectral clustering,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, no. 4, p. 17, 2009.

[66] K. S. Xu, M. Kliger, and A. O. Hero, “Evolutionary spectral clustering with adaptive forgetting factor,” in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010, pp. 2174–2177.

[67] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang, “Incremental spectral clustering with application to monitoring of evolving blog communities,” in *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 261–272.

[68] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, “Community evolution in dynamic multi-mode networks,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 677–685.

[69] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, “Analyzing communities and their evolutions in dynamic social networks,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, no. 2, p. 8, 2009.

[70] J. Zhang, Y. Song, G. Chen, and C. Zhang, “On-line evolutionary exponential family mixture,” in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[71] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, “Community structure in time-dependent, multiscale, and multiplex networks,” *science*, vol. 328, no. 5980, pp. 876–878, 2010.

[72] J. Rosswog and K. Ghose, “Detecting and tracking spatio-temporal clusters with adaptive history filtering,” in *2008 IEEE International Conference on Data Mining Workshops*. IEEE, 2008, pp. 448–457.

[73] K. S. Xu, M. Kliger, and A. O. Hero Iii, “Adaptive evolutionary clustering,” *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 304–336, 2014.

[74] S. M. Smith and J. M. Brady, “Asset-2: Real-time motion segmentation and shape tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 814–820, 1995.

[75] R. T. Collins, Y. Liu, and M. Leordeanu, “Online selection of discriminative tracking features,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 10, pp. 1631–1643, 2005.

[76] A. Vahdat, M. Heywood, and N. Zincir-Heywood, “Bottom-up evolutionary subspace clustering,” in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[77] A. Vahdat, M. I. Heywood, and A. N. Zincir-Heywood, “Symbiotic evolutionary subspace clustering,” in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[78] A. Vahdat and M. I. Heywood, “On evolutionary subspace clustering with symbiosis,” *Evolutionary Intelligence*, vol. 6, no. 4, pp. 229–256, 2014.

[79] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.

[80] “Deep learning toolbox.” [Online]. Available: https://au.mathworks.com/help/deeplearning/

[81] “Inputsize.” [Online]. Available: https://au.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.sequenceinputlayer.html

[82] “Numhiddenunits.” [Online]. Available: https://au.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.lstmlayer.html

[83] “Outputsize.” [Online]. Available: https://au.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html

[84] C. Tomasi and T. Kanade, “Shape and motion from image streams under orthography: a factorization method,” *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[85] “Jhu johns hopkins computer vision machine learning.” [Online]. Available: http://www.vision.jhu.edu/

[86] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[87] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, “Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition,” in *Proceedings of 27th Asilomar conference on signals, systems and computers*. IEEE, 1993, pp. 40–44.

[88] A. Hashemi and H. Vikalo, “Accelerated sparse subspace clustering,” *arXiv preprint arXiv:1711.00126*, 2017.

[89] ——, “Sparse linear regression via generalized orthogonal least-squares,” in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2016, pp. 1305–1309.

[90] S. Swaroop, C. V. Nguyen, T. D. Bui, and R. E. Turner, “Improving and understanding variational continual learning,” *arXiv preprint arXiv:1905.02099*, 2019.