SPARSE SUBSPACE CLUSTERING FOR EVOLVING DATA STREAMS

Jinping Sui [1,2], *Zhen Liu* [1], *Li Liu* [3,4], *Alexander Jung* [2], *Tianpeng Liu* [1], *Bo Peng* [1], *Xiang Li* [1]

[1]College of Electronic Science, National University of Defense Technology, Changsha, China 410073
[2]Department of Computer Science, Aalto University, Espoo, Finland 02150
[3]College of System Engineering, National University of Defense Technology, Changsha, China 410073
[4]Center of Machine Vision and Signal Analysis, University of Oulu, Finland

ABSTRACT

The data streams arising in many applications can be modeled as a union of low-dimensional subspaces known as multi-subspace data streams (MSDSs). Clustering MSDSs according to their underlying low-dimensional subspaces is a challenging problem which has not been resolved satisfactorily by existing data stream clustering (DSC) algorithms. In this paper, we propose a sparse-based DSC algorithm, which we refer to as dynamic sparse subspace clustering (D-SSC). This algorithm recovers the low-dimensional subspaces (structures) of high-dimensional data streams and finds an explicit assignment of points to subspaces in an online manner. Moreover, as an online algorithm, D-SSC is able to cope with the time-varying structure of MSDSs. The effectiveness of D-SSC is evaluated using numerical experiments.

***Index Terms*—** Data stream clustering, high-dimensional data stream, subspace clustering, online clustering

1. INTRODUCTION

In recent years, high-dimensional data streams have been continuously generated at an unprecedented rate in various fields such as media, communication, finance, meteorology, etc. [1, 2, 3, 4]. These data streams often feature high dimensionality, no labeling, massiveness, and evolving, presenting huge challenges to data stream clustering (DSC) algorithms. On one hand, as the dimensionality increases, most existing DSC algorithms perform poorly, commonly known as curse of dimensionality [5]. On the other hand, the constant evolution of the data stream easily results in previously effective models no longer being applicable [6, 7]. Therefore, how to deal with high dimensional data streams especially the ones with evolving property is a major problem that plagues the development of the field.

It has been realized that most high-dimensional data streams actually lie in a union of low-dimensional subspaces rather than uniformly distribute in the whole ambient space [1, 2, 8]. These data streams are referred to as multi-subspace data streams (MSDSs) in this paper. A classical example of MSDSs in real life is the collection of front face images. It has been observed that the front face images of a subject with a fixed facial expression and varying illumination lie close to a linear subspace of dimension 9 under the Lambertian assumption [2], which implies the collection of face images of multi-subjects lies close to a union of 9D subspaces [1]. Therefore, finding the latent multi-subspace and clustering the high-dimensional data streams according to their original subspaces provides us a possible way to reduce the computational complexity and storage resource consumption.

Prior Art. Although MSDS has been generated in various fields in practice, research on its processing is still in an immature stage. One of the main reasons is that in the high-dimensional space all pairs of points tend to be almost equidistant from one another [9]. Therefore, most partitioning based DSC methods such as CluStream [10], STRAP [11] cannot keep their good performance any more when processing high-dimensional data streams. Yet only a few approaches have been proposed so far to tackle high dimensional data stream clustering problem. They can be largely divided into two categories: density-based DSC algorithms and projection-based DSC algorithms. Density-based algorithms, such as DenStream [12], CEDAS [13], has been verified on real MSDSs. However, it should be noted that these density-based algorithms are full dimensional processing methods. As the data dimension increases further, their accuracy and computational complexity will be severely adversely affected. Projection-based DSC algorithms, such as HPStream [9], HDDStream [14], can search clusters from subspaces. However, these methods rely much on prior information about the subspaces, which is not easy to know in advance [3, 14]. Moreover, the evolving property of data streams is not considered enough by these mentioned methods. Generally, they only focus on detecting the new subspaces (clusters) while ignoring the subspace disappearance or recurrence.

Contributions. In this paper, a novel DSC algorithm is proposed, named D-SSC, which can cluster MSDSs in an online manner. D-SSC has two stages. The first is online clustering the arriving data according to the current D-SSC summary. Different from most existing DSC techniques, D-

SSC is not based on the distance or density similarity while taking advantage of a unique property of MSDSs called *self-expressiveness i.e.,* each data point in a union of subspaces can be efficiently represented as linear (affine) combinations of other points. Among these combinations, a sparse representation corresponds to a combination of points from its own subspace [1, 8]. This motivates us to find a subspace for a data point by solving a global sparse optimization problem. The second stage is D-SSC summary online updating. D-SSC summary stores global information of the data stream and local information of each discovered subspace up to the current timestamp. Considering that most of the data streams in practice have evolving characteristics, an evolution detection strategy investigated under which the evolving cases, such as emerging, disappearing, recurring of subspaces, can be detected. This ensures that the proposed algorithm can more effectively reflect the current pattern of the MSDSs.

2. PROBLEM FORMULATION

2.1. MSDS Clustering Problem and Self-expressiveness Property

A MSDS is defined as a sequence of data points which lies in a union of low-dimensional subspaces. Different from batch multi-subspace datasets [15], a unique property of MSDSs is we can only access its data points in an online manner. Assume that at each timestamp t, we receive the data point $\boldsymbol{x}^t$ and $\boldsymbol{x}^t \in \mathbb{R}^{D\times 1}$. We denote the t points which we have received as $\mathbf{X}^t = [\boldsymbol{x}^1 \dots \boldsymbol{x}^t]_{D\times t}$. Generally, the MSDS clustering problem is to dynamically identify k^t subspaces $\mathbb{S}^t = \{\mathcal{S}_l^t\}_{l=1}^{k^t}$ at timestamp t and find an appropriate subspace to assign $\boldsymbol{x}^t$ into. Each $\mathcal{S}_l^t$ is a d_l^t-dimensional subspace and contains N_l^t data points. It should be pointed out that $N_l^t > d_l^t$ holds here for a MSDS [1].

In order to solve the MSDS subspace clustering problem, a unique property for multi-subspace data points is utilized in this work, that is, *self-expressiveness* property [1, 8, 16].

Self-expressiveness Property: For N data points $\mathbf{X}$ from multi-subspace, each data point $\boldsymbol{x}_i$ can be represented as a linear or affine combination of other points, that is,

$$\boldsymbol{x}_i = \mathbf{X}\boldsymbol{c}_i, \;\; c_{ii} = 0, \tag{1}$$

where $\boldsymbol{c}_i \triangleq [c_{i1}c_{i2} \dots c_{iN}]^\top$ and $c_{ii} = 0$ avoid an extreme solution of expressing $\boldsymbol{x}_i$ by itself. Obviously, $\boldsymbol{c}_i$ is not unique under the assumption of $N_l^t > d_l^t$ [1]. However, it has been proved that there exists a sparse solution of $\boldsymbol{c}_i$ whose nonzero entries only corresponds to a few points from the same subspace as $\boldsymbol{x}_i$. Therefore, this sparse solution provides us a way to find some points in the same subspace as $\boldsymbol{x}_i$.

For all data points $\mathbf{X}$ we have

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

where $\mathbf{C} \triangleq [\boldsymbol{c}_1 \boldsymbol{c}_2 \dots \boldsymbol{c}_N] \in \mathbb{R}^{N\times N}$ is referred as *subspace self-representation matrix* in this paper whose ith column corresponds to the sparse representation of data point $\boldsymbol{x}_i$. $\text{diag}(\mathbf{C})$ is the vector of the diagonal elements of $\mathbf{C}$.

2.2. Subspace Evolution

We have to consider the time-evolving property of the subspaces of MSDSs due to the fact the points of MSDSs are accessed and processed dynamically. The time-evolving property here refers to the case that the MSDSs data points evolve over time in the distribution of subspaces, which is commonly referred to as *virtual concept drift* in the field of DSC research [17]. The MSDSs with subspace evolution is referred to as *evolving MSDSs* in this paper. More precisely, we consider the three most possible subspace evolution cases in this work, *i.e., subspace emergence, disappearance, and reoccurrence.* The definitions are given as follows.

Subspace Emergence. Subspace emergence refers to the occurrence of a new subspace at timestamp t. In particular, a subspace $\mathcal{S}$ emerges at timestamp t if $\mathcal{S} \notin \mathbb{S}^1 \cup \mathbb{S}^2 \cup \dots \cup \mathbb{S}^{t-1}$ and $\mathcal{S} \in \mathbb{S}^t$.

Subspace Disappearance. Subspace disappearance is defined as an existing subspace that is not visited by the recently arrived data points. Formally, a subspace $\mathcal{S}$ disappears if $\mathcal{S} \in \mathbb{S}^{t_0} \cap \mathbb{S}^{t_0+1} \cap \dots \cap \mathbb{S}^{t-1}$ and $\mathcal{S} \notin \mathbb{S}^t$, where $1 \le t_0 < t$.

Subspace Reoccurrence. Subspace reoccurrence means the situation where a previously disappeared subspace recurs at timestamp t. Formally, a subspace $\mathcal{S}$ recurs at timestamp t if $\mathcal{S} \in \mathbb{S}^{t_1} \cap \mathbb{S}^{t_1+1} \cap \dots \cap \mathbb{S}^{t_2-1}$, $\mathcal{S} \notin \mathbb{S}^{t_2} \cup \mathbb{S}^{t_2+1} \cup \dots \cup \mathbb{S}^{t-1}$, and $\mathcal{S} \in \mathbb{S}^t$, where $1 \le t_1 < t_2 < t$.

3. MULTI-SUBSPACE DATA STREAM CLUSTERING ALGORITHM

In this section, we will give a specific demonstration about D-SSC algorithm. The D-SSC summary structure and its initialization are given firstly. Then, we explain the online clustering and evolution detection of D-SSC.

3.1. The D-SSC Summary Structure and Its Initialization

As an online subspace clustering algorithm, D-SSC is capable of finding the underlying low-dimensional subspaces and assigning all received data points of MSDSs. The subspace clustering result $\mathbb{S}^t = \{\mathcal{S}^t\}_{l=1}^{k^t}$ which we refer to as *D-SSC* summary, is recorded and updated in an online manner. Each $\mathcal{S}_l^t = \{n_l^t, \mathbf{R}_l^t, t_l, p_l^t, q^t\}$ is a 5-tuple which summarizes the information for the lth subspace, where

•n_l^t is the total number of data points assigned to subspace l up to timestamp t;

•$\mathbf{R}_l^t$ is referred to as *reserved data matrix* of subspace l which saved some selected points from subspace l up to timestamp t;

•t_l is the last timestamp when a point was assigned to subspace l;

•p_l^t is a counter whose initial value is set as 0;

•q^t is the total number of outliers found in the MSDSs up to timestamp t. It should be noted that for all subspaces, q^t holds equally.

D-SSC needs to be initialized firstly. Hence, the first batch of $M(M > D)$ arriving data points $\mathbf{X}^M$ will be utilized to find the subspaces underlying $\mathbf{X}^M$. Meanwhile, we expect the most informative points of each subspace can be reserved in the D-SSC summary for further processing due to the limitation of storage space. Hence, the number of nonzero rows of $\mathbf{C}^M$ is also minimized when we take $\mathbf{X}^M$ into (2) to solve the subspace self-representation matrix $\mathbf{C}^M$. We use the term $\|\mathbf{C}^M\|_{r,0} \triangleq \sum_{i=1}^{M} I(\|\boldsymbol{c}^i\|_2)$ to characterize the number of nonzero rows of matrix $\mathbf{C}^M$, where $\boldsymbol{c}^i$ denotes the ith row of $\mathbf{C}^M$ and $I(\cdot)$ denotes the indicator function. Hence, an ideal $\mathbf{C}^M$ can be got by solving the following optimization problem,

$$\min \|\mathbf{C}^M\|_0 + \lambda \|\mathbf{C}^M\|_{r,0} \text{ s.t.} \mathbf{X}^M = \mathbf{X}^M \mathbf{C}^M, \text{diag}(\mathbf{C}^M) = 0, \tag{3}$$

where $\lambda > 0$ is a parameter to adjust the balance between the sparsity and the number of nonzero row of the solution. However, (3) is a NP-hard problem. Hence, we consider a convex relaxation of (3).

$$\min \|\mathbf{C}^M\|_1 + \lambda \|\mathbf{C}^M\|_{r,1} \text{ s.t.} \mathbf{X}^M = \mathbf{X}^M \mathbf{C}^M, \text{diag}(\mathbf{C}^M) = 0, \tag{4}$$

where $\|\mathbf{C}^M\|_{r,1} \triangleq \sum_{i=1}^{M} \|\boldsymbol{c}^i\|_2$. It has been verified under independent subspaces assumption, the solution of (4), denoted as $\widehat{\mathbf{C}}^M$, perfectly contains informative points from each underlying subspaces [18]. Now assume that $\widehat{\mathbf{C}}^M$ has b^M nonzero rows which correspond to the *informative points*. Moreover, $\widehat{\mathbf{C}}^M$ can be further used to infer the clustering of $\mathbf{X}$ by spectral clustering methods, which is not the focus of this work. We just assume that the clustering result, *i.e.,* the assignment for M points into l^M subspaces has been gained. Thereby, we can record the n_l^M and t_l information for each subspace. The b^M informative points are reserved in the D-SSC summary for further processing which will explained in Section 3.2. It should be pointed out that b^M is generally less than M. Therefore, in order to build an over complete data matrix for further processing, we need to guarantee the number of points reserved, expressed as M_0, greater than D if we want to utilize the self-expressiveness property. Thus, we then randomly select a_l^M points from those non-informative points in each subspace,

$$a_l^M = \lceil ((M_0 - b^M) \frac{n_l^M}{M}) \rceil, \tag{5}$$

where $\lceil \cdot \rceil$ is the ceiling function. Thus we also save these points in corresponding $\mathbf{R}_l^M$.

3.2. Online Clustering and Evolution Detection

Depending on whether the subspaces can effectively represent the current pattern of MSDSs, D-SSC divides the subspaces into two states, *i.e., active and inactive,* at each timestamp. Inactive state means the corresponding subspaces have been expired. Both states can be converted to each other over time. D-SSC does not directly delete inactive subspaces but stores them to another reservoir $\mathbb{D}^t = \{\mathcal{D}_i^t\}_{i=1}^{h^t}$ which we refer to as *remove reservoir*. Here we assume that at timestamp t, there are h^t inactive subspaces in $\mathbb{D}$. Due to the active and inactive can be converted to each other, the inactive subspace can be denoted as $\mathcal{D}_i^t = [\widetilde{n}_i^t, \widetilde{\mathbf{R}}_i^t, \widetilde{t}_i, \widetilde{p}_i^t, \widetilde{q}^t]$. Meanwhile, we define $\mathbf{Z}^t = [\mathbf{R}_1^t \cdots \mathbf{R}_{k^t}^t \widetilde{\mathbf{R}}_1^t \cdots \widetilde{\mathbf{R}}_{h^t}^t]$ here. $\mathbf{Z}^t$ is made up by all reserved data matrix of active and inactive subspaces at timestamp t.

For a new arriving data point $\boldsymbol{x}^t (t > M)$, we firstly get its representative coefficients under $\mathbf{Z}^{t-1}$, that is,

$$\min \| \boldsymbol{c}^t \|_0 \quad \text{s.t. } \boldsymbol{x}^t = \mathbf{Z}^{t-1} \boldsymbol{c}^t. \tag{6}$$

We can also use ℓ_1-norm to relax (6) to get the solution $\widehat{\boldsymbol{c}}^t$. According to the self-expressiveness property, we know that $\widehat{\boldsymbol{c}}^t$ has block sparsity property if $\boldsymbol{x}^t$ is from a found subspace (we call such $\boldsymbol{x}^t$ a normal point). That is, non-zero elements of $\widehat{\boldsymbol{c}}^t$ are concentrated in a certain part (this part corresponds to the reserved matrix of its own subspace). Hence, we calculate the sparsity concentration index (SCI) [8] of $\widehat{\boldsymbol{c}}^t$ by

$$\text{SCI}(\widehat{\boldsymbol{c}}^t) \triangleq \frac{(k^{t-1} + h^{t-1}) \cdot \max_j (\| \delta_j(\widehat{\boldsymbol{c}}^t) \|_1 / \| \widehat{\boldsymbol{c}}^t \|_1) - 1}{(k^{t-1} + h^{t-1}) - 1}, \tag{7}$$

where $\delta_j(\widehat{\boldsymbol{c}}^t)$ is a function that selects the coefficients associated with the jth ($j \in [1, k^{t-1} + h^{t-1}]$) subspace in $\widehat{\boldsymbol{c}}^t$. $\text{SCI}(\widehat{\boldsymbol{c}}^t) \in [0, 1]$ and a higher $\text{SCI}(\widehat{\boldsymbol{c}}^t)$ means the coefficients of $\widehat{\boldsymbol{c}}^t$ is more likely to concentrate on a single subspace. Hence, we introduce a threshold τ and accept $\boldsymbol{x}^t$ as a normal point if

$$\text{SCI}(\widehat{\boldsymbol{c}}^t) \geqslant \tau, \tag{8}$$

and otherwise reject as an *outlier*. The outlier will be saved in an outlier reservoir, denoted as $\mathcal{O}^t$. For a normal point, we update n_l^t, t_l in its corresponding $\mathcal{S}_l$ if it corresponds to an active subspace, or update $\widetilde{n}_i^t, \widetilde{t}_i, \widetilde{p}_i^t$, and otherwise we update q^t in the D-SSC summary. It should be mentioned the updating rule for $\widetilde{p}_i^t$ is $\widetilde{p}_i^t = \widetilde{p}_i^{t-1} + 1$.

D-SSC can deal with evolving MSDSs by detecting three possible subspace evolution, *i.e.,* subspace emergence, disappearance and reoccurrence. When new subspace(s) emerges, its points will be rejected by D-SSC as outliers. Hence, at each timestamp, D-SSC checks if qt exceeds an emergence detection threshold $\alpha (\alpha > N)$. If $q^t \geqslant \alpha$, (3)-(5) will be applied to the data matrix of all points in $\mathcal{O}^t$ to find the new subspaces and their summaries. For each active subspace l we calculate the time interval between the last timestamp when it was visited and the current timestamp, *i.e.,*

$$\Delta_{tl} = t - t_l. \tag{9}$$

Then, Δ_{tl} is compared with a threshold β to judge the state of subspace l at timestamp t. The subspace will be rejected as active subspace when $\Delta_{tl} \geqslant \beta$. A common feature that most practical MSDSs share is some inactive subspaces will

recur after a long time interval. For an inactive subspaces, $\widetilde{p}_i^t$ will be compared with a threshold γ at each timestamp. An incative subspace will be accepted as an active subspace if $\widetilde{p}_i^t \geqslant \gamma$.

4. EXPERIMENTS RESULTS

In this section, the performance of the proposed algorithm is tested on several real MSDSs. Three related and state of the art algorithms, STRAP [11], CEDAS [13], SSC [1] are selected of which results are compared with those of D-SSC. It should be noted that SSC is designed only for batch datasets subspace clustering. Here we feed the tested data stream as a whole batch dataset into SSC to get its results. The four realistic MSDSs are generated from USPS [19] and MNIST [20] datasets shown in Tabel 1. Note the datasets has been applied RPCA algorithm [21] to remove the sparse outlying entries of each subspace. We enhance the evolving property of the MSDSs by rearranging the order of access of their points.

Table 1. MSDSs used for experiments.

	features	samples	subspaces
USPS	256	9298	10
MNIST-10K	784	10000	10
MNIST-15K	784	15000	10
MNIST-20K	784	20000	10

We compare the results of each algorithm from three perspectives, namely the accuracy of the subspace clustering, the number of subspaces that are actually found, and the number of all subspaces generated. The number of all subspaces generated is the total number of clusters in the clustering results while the number of subspaces found means how many different subspaces are found compared with the ground truth when using the label information of data points. The results recorded in Tabel 2. As demonstrated in Table 2, D-SSC outperforms the other state of the art algorithms with much higher subspace clustering accuracy and more efficient in terms of finding the underlying subspaces, i.e., finding more subspaces with less subspaces generating. For each MSDS, CEDAS and STRAP can only find a few subspaces and split one subspace into a lot of parts. The reason is that the traditional distance measurements lose their effectiveness in the high-dimensional spaces, which resulting in points from different subspaces are pretty near to each other. Hence, they cannot distinguish the subspaces which are near in the full feature space, thus some actually different subspaces are regarded as the same one. Compared with D-SSC, SSC is less effective in subspace clustering accuracy. This is mainly because the performance of the SSC is greatly affected as the number of data, dimensions, and number of subspaces increase. As an online processing method, D-SSC can avoid the negative impact of directly processing an large dataset.

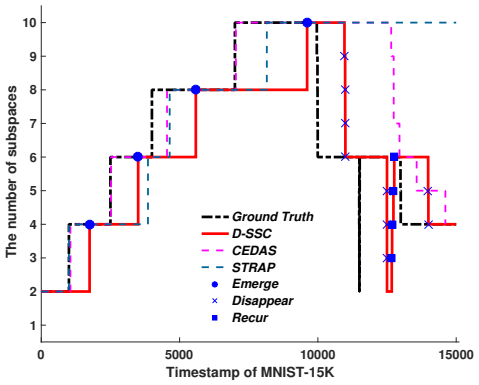

Fig. 1. The results of D-SSC, CEDAS, and STRAP on MNIST-15K MSDSs.

Table 2. Subspace clustering accuracy (%), the number of subspaces found (*N1*), and the number of subspaces generated (*N2*). The corresponding results are demonstrated as *'N1/N2'* in the table.

	D-SSC	CEDAS	STRAP	SSC
USPS	**59.32**	32.38	38.85	47.85
	10/10	7/13	6/156	10/10
MNIST-10K	**55.19**	32.73	37.00	43.77
	10/10	5/68	6/122	10/10
MNIST-15K	**53.95**	30.55	35.72	40.38
	10/10	10/106	10/136	10/10
MNIST-20K	**52.15**	27.76	33.59	37.91
	10/10	3/76	6/82	10/10

In order to demonstrate the tracking performance of D-SSC on subspace evolution, we further demonstrate the performance of D-SSC in terms of tracking the subspace evolution in Fig. 1. We also calculate the number of subspaces found at each timestamp of CEDAS and STRAP. From Fig. 1, we can find that D-SSC can track the three types of subspace evolution successfully. While CEDAS and STRAP can track the subspace emergence as well, they always generate hundreds of subspaces in reality.

5. CONCLUSIONS

In this paper, we propose an efficient data stream clustering algorithm, named D-SSC, which can recover the underlying subspaces of evolving MSDSs and find a unique assignment of the points to subspaces. The typical evolution of subspaces, such as subspace emergence, disappearance, and reoccurrence can be detected and processed in real time which ensures MSDSs is more practical and effective when process MSDSs in real world. The effectiveness and superiority of proposed algorithm have been verified by numerical experiments.

6. REFERENCES

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

[2] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 218–233, Feb 2003.

[3] Xin Jiang Hunt and Rebecca Willett, "Online data thinning via multi-subspace tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[4] Hai-Long Nguyen, Yew-Kwong Woon, and Wee-Keong Ng, "A survey on data stream clustering and classification," *Knowledge and information systems*, vol. 45, no. 3, pp. 535–569, 2015.

[5] Richard E Bellman, *Adaptive control processes: a guided tour*, vol. 2045, Princeton university press, 2015.

[6] Conor Fahy, Shengxiang Yang, and Mario Gongora, "Ant colony stream clustering: A fast density clustering algorithm for dynamic data streams," *IEEE Transactions on Cybernetics*, 2018.

[7] Yu Sun, Ke Tang, Leandro L Minku, Shuo Wang, and Xin Yao, "Online ensemble learning of data streams with gradually evolved classes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1532–1545, 2016.

[8] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma, "Robust face recognition via sparse representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[9] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu, "A framework for projected clustering of high dimensional data streams," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 852–863.

[10] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[11] Xiangliang Zhang, Cyril Furtlehner, Cecile Germain-Renaud, and Michele Sebag, "Data stream clustering with affinity propagation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1644–1656, 2014.

[12] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM international conference on data mining*. SIAM, 2006, pp. 328–339.

[13] Richard Hyde, Plamen Angelov, and Angus Robert MacKenzie, "Fully online clustering of evolving data streams into arbitrarily shaped clusters," *Information Sciences*, vol. 382, pp. 96–114, 2017.

[14] Irene Ntoutsi, Arthur Zimek, Themis Palpanas, Peer Kröger, and Hans-Peter Kriegel, "Density-based projected clustering over high dimensional data streams," in *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM, 2012, pp. 987–998.

[15] Yigit Oktar and Mehmet Turkan, "A review of sparsity-based clustering methods," *Signal Processing*, 2018.

[16] Xiaozhao Fang, Na Han, Wai Keung Wong, Shaohua Teng, Jigang Wu, Shengli Xie, and Xuelong Li, "Flexible affinity matrix learning for unsupervised and semisupervised classification," *IEEE transactions on neural networks and learning systems*, , no. 99, pp. 1–17, 2018.

[17] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[18] Ehsan Elhamifar, Guillermo Sapiro, and Rene Vidal, "See all by looking at a few: Sparse modeling for finding representative objects," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 1600–1607.

[19] Jonathan J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 11, 2011.