Received December 31, 2020, accepted January 11, 2021, date of publication April 14, 2021, date of current version April 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3054767

Sparse Subspace Clustering for Stream Data

KEN CHEN[1], YONG TANG[1], LONG WEI[2], PENGFEI WANG[2], YONG LIU[3], AND ZHONGMING JIN[2]
[1]Sichuan Railway Industry Investment Group Company Ltd., Chengdu 610094, China
[2]Alibaba DAMO Academy, Alibaba Group, Hangzhou 310024, China
[3]Alibaba Cloud, Alibaba Group, Hangzhou 310024, China

Corresponding author: Ken Chen (chenken@foxmail.com)

ABSTRACT In the past few years, sparse subspace clustering (SSC) has gained many studies and found wide applications. However, SSC suffers from the limitation in scalability. Furthermore, current SSC methods could hardly tackle stream data where the structure of subspaces may change along time. In this paper, we propose a novel method to extend SSC to stream data (StreamSSC). Our method is based on maintaining a small subset of representatives to characterize the structure of the underlying subspaces during stream data. StreamSSC is efficient in both computation and memory. Experimental results on both synthetic and real-world streams demonstrate the effectiveness of StreamSSC. For efficiency, StreamSSC is faster than existing online subspace clustering methods by roughly a magnitude.

INDEX TERMS Clustering algorithms, sparse subspace clustering, stream data.

I. INTRODUCTION

In many real-world applications, samples from different classes approximately lie in a union of multiple low dimensional subspaces. The subspace clustering problem aims to segment samples into multiple clusters based on their underlying subspaces. In the past few years, it has attracted increasing research interest and found wide applications in face clustering [1], handwritten digit clustering [2] and motion segmentation [3], [4]. There have been various subspace clustering algorithms [2], [5], [6] proposed to segment data into multiple clusters based on subspace structure.

Among these algorithms, sparse subspace clustering (SSC) [1] is currently the most popular method, due to its theoretical guarantee [5] and impressive performance in practice [6]. Formally, SSC first learns the sparse representations C of all data points by solving the following optimization program:

$$\min_{C} \|C\|_1 \quad s.t.\, X = XC,\ \operatorname{diag}(C) = 0, \tag{1}$$

where X is the matrix of points, and each column of X corresponds to one data point. Then, it applies spectral clustering on the affinity matrix $A = |C| + |C|^T$ to divide the data into multiple clusters.

According to Eq. 1, SSC needs to access all points in memory, and compute the sparse representations for all data points in X. The expensive memory and time consumption severely limits its scalability to stream data in which the size of data is quite huge and even infinite. Although there have been some scalable solvers proposed recently [7]–[11], they cannot tackle the situation under which points in the same clusters may arrive at different times and so the structure of underlying subspaces changes over time. As a result, there is a need to develop a flexible and efficient SSC method for stream data.

The only existing attempt towards subspace clustering for stream data is online low-rank subspace clustering by basis dictionary pursuit (OLRSC) [12], which is an online implementation of LRR via a stochastic optimization scheme, while the counterpart for SSC remains open. Meanwhile, recent work [8], [13] shows that SSC could obtain subspace-preserving affinity with the less restrictive assumption on the distribution of samples compared with low rank representation (LRR). As a result, there is a need to develop a flexible and efficient SSC method for stream data.

However, we could hardly apply SSC directly to stream data, due to the following challenges: (1) it needs to learn the sparse representation of each point over all the remaining points. The size of stream data is typically very large, thus the accessibility to the entire stream to compute linear combinations is impractical. It is also prohibitive to store the entire stream in memory. (2) Clustering methods for stream data require real-time clustering results of each new arriving point, thus assigning the new point to its underlying subspace should be implemented extremely fast. (3) A more essential

The associate editor coordinating the review of this manuscript and approving it for publication was Xiangtao Li.

difficulty comes from the nature of stream data, where the structure of subspaces is changing gradually along time, one needs to detect these changes accordingly, but there is no variant of SSC up to now that could capture these dynamics.

In this paper, we propose a novel sparse subspace clustering method for stream data (StreamSSC). In our method, we keep track of the potential change in the structure of subspaces during the stream. Meanwhile, we select and maintain a representative set to characterize the structure of observed subspaces. For a new arriving data point, we seek its sparse linear combination over the representative set, upon which we will investigate possible change taking place in the subspace structure. When the change is detected, we update the subspaces by performing SSC on a small subset of points. Then a new structure of subspaces will be obtained and a new representative set will be selected to replace the previous ones, respectively. Our clustering framework repeats this ''subspace structure change detection - subspace structure update'' procedure until the end of the stream. Finally, the experimental results on both synthetic and real-world streams demonstrate the effectiveness and efficiency of our method.

Our method inherits the merits of the SSC model, while overcomes all the above-mentioned challenges in stream data. Specifically, our paper has the following contributions:

- We propose an effective strategy to select the representative set with minimal sampling complexity. StreamSSC is capable of predicting the real-time clustering result of each point as well as capturing changes in the structure of subspaces.
- StreamSSC is efficient in both computation and memory. The memory cost is only $O(Kd)$, where K is the total number of subspaces emerging in the stream and d is an upper bound of dimensions of these subspaces. Experimental results show that StreamSSC is faster than existing online subspace clustering methods by roughly a magnitude.

This paper is organized as follows. In section 2, we review related work. Then we propose our sparse subspace clustering method for stream data in section 3. In section 4, experiments are conducted on both synthetic and real-world streams. The last section concludes this paper.

II. RELATED WORK

A. SUBSPACE CLUSTERING

In the past several years, there are two popular subspace clustering approaches: sparse subspace clustering (SSC) [1] and low rank representation (LRR) [2], which have received intensive study [2], [5], [6]. Both SSC and LRR segment datasets based on self-expressiveness among data points. Formally, SSC and LRR aim to segment samples into their underlying subspaces by solving

$$\min_{C} \|C\| \quad s.t.\, X = XC,\ diag(C) = 0 \tag{2}$$

where X is the matrix of points. Then spectral clustering is applied to the affinity matrix $A = |C|+|C|^T$ to cluster points. The difference between SSC and LRR is that the norm $\|.\|$ of C is chosen as l_1 in SSC while the nuclear norm in LLR, to encourage sparsity and low-rank property, respectively.

From then on, some variants of SSC and LRR have been proposed to handle data under different scenes, such as Low Rank Subspace Clustering [14], [15], Low Rank Sparse Subspace Clustering [13], Latent LLR [16] and Structured SSC [17], to name a few. There are also following work focusing on certain aspects of subspace clustering, such as out of sample problem [18], samples with missing entries [19], outlier detection [5], dimensionality reduction [20] and graph connectivity [21]. However, most of these methods suffer from high time and memory complexities. Thus they are not suitable for stream data whose size is typically quite large.

B. DISCUSSION ABOUT SCALABLE SUBSPACE CLUSTERING

In recent years, many methods are proposed to improve the scalability of SSC. Some scalable SSC solvers [7], [18] suggest to utilize a small subset to represent the entire dataset to reduce the computational cost. The recently proposed Scalable Sparse Subspace Clustering by Orthogonal Matching Pursuit (OMP-SSC) [8] adopts orthogonal matching pursuit to SSC and works well on massive data.

For example, Chen *et al.* [10] propose a dropout technique to address the issue of over-segmentation that appeared in current SSC methods and improve their scalability. Motivated by Robust Principal Component Analysis (RPCA), Zhang *et al.* design an Adaptive Low-rank Model for subspace clustering [9]. To achieve the same goal, Matsushima and Brbic [11] propose a Selective Sampling-based Scalable Sparse Subspace Clustering (S5C) algorithm. For high-dimensional data, random sketching approaches are adopted to reduce data dimensionality and thus achieves acceleration [22].

Although the above scalable subspace clustering methods show appealing performance, they could hardly be applied to subspace clustering on stream data. The reason is that these method segments the whole static dataset directly. Two methods could be applied to stream data. The first one is OLRSC [12], which extends LRR via a stochastic optimization scheme. The second one is online robust PCA via stochastic optimization (ORPCA) [23], which recovers low-rank subspace structure incrementally. Then the learned low dimensional representations of all points can be applied to perform clustering [12]. However, they could not output real-time clustering results of each new arriving point, nor could they detect subspace structure change in a stream, which is one of the most challenging difficulties in stream data clustering.

Beyond data forms of vectors, Li *et al.* consider tensor data (i.e., the data having multiple dimensions) and propose a method of online robust low-rank tensor modeling for streaming data analysis [24].

III. SPARSE SUBSPACE CLUSTERING FOR STREAM DATA

In this section, we introduce our Sparse Subspace Clustering method for Stream data (StreamSSC). We start with the problem setup and an overview of our framework, then we will address several important technical details of our method.

A. PROBLEM SETUP

Consider a data stream $X = \{x_t\}_{t=1}^{\infty}$ in which all points are sampled from a union of K low dimensional linear subspaces $\mathcal{S}_1, \cdots, \mathcal{S}_K \subset \mathbb{R}^D$, and the dimension of each subspace $\mathcal{S}_i$ is $d_i \ll D$. Points that lie in the same subspace $\mathcal{S}_i$ are regarded as a cluster C_i. In the context of stream, points from the same subspace might arrive at different times. Thus, the subspace structure involved in the stream may change gradually over time. All the points are normalized to have unit l_2 norm. In this paper, we aim to solve the clustering problem over such a massive stream via SSC efficiently.

For stream data, it is impractical to explore the exact subspace structure via SSC. As a result, researchers usually leverage a subset of points sampled from the stream to get an approximate solution [25]. Therefore, we introduce an important definition, *representative set*, as follows.

Definition 1: Suppose there are k subspaces $\{\mathcal{S}_i^{(t)}\}_{i=1}^{k}$ *observed over X at time t,* $\mathcal{R}_i^{(t)} = \{r_i^1, \cdots, r_i^{n_i}\} \subset X$ *is defined as a* representative set *of subspace* $\mathcal{S}_i^{(t)}$ *if* $span(\mathcal{R}_i^{(t)}) = \mathcal{S}_i^{(t)}$. *Each point in* $\mathcal{R}_i^{(t)}$ *is called a* representative. $\mathcal{R}^{(t)} = \bigcup_{i=1}^{k} \mathcal{R}_i^{(t)}$ *is called the* universal representative set.

We use lowercase k to denote the current number of clusters, which is dynamic along with coming of new data points. And we use uppercase K to denote the number of underlying clusters during the whole stream, which is a fixed number.

From the definition, the representative set is a summary of each subspace because the representative set spans the individual subspace and thus serves as an over-complete basis. The utilization of representatives overcomes the difficulty of seeking linear combination over all points. Note that we do not require representatives from the same subspace to be linear independent because the intrinsic subspace dimensions d_i's are usually unknown, and this relaxes our selection of representatives to be more flexible.

At any time during our clustering process, we use the representative sets to describe the current clustering structure. We emphasize that the representative sets may change along with the variation of the cluster structure. How to select and maintain the representatives sets will be detailed in the following section.

B. FRAMEWORK OVERVIEW

The basic idea of our method is exploiting the representative set to perform subspace clustering on stream data efficiently in both computation and memory, then changes in the structure of subspaces are captured by the outlier detection technique.

We present an overview of our framework in Figure 1. At the beginning stage, we collect T_0 points $X_0 = \{x_1, \cdots, x_{T_0}\} \subset X$ to perform standard SSC [1] as initialization. We adopt the self-tuned strategy [26] to determine the number of clusters. Then we select a small subset of points from each cluster to act as a representative set for the corresponding subspace.

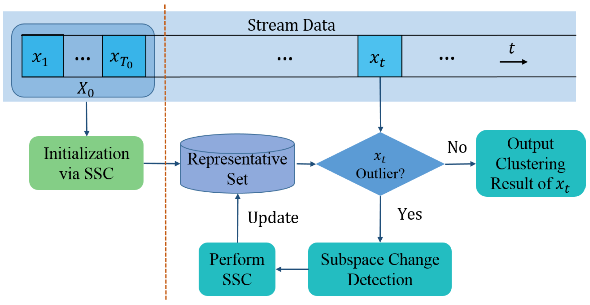


FIGURE 1. Overview of our StreamSSC framework. During the stream data, we select and maintain a small subset of points as the representative set. For each new arriving point x_t, we decide whether it is an outlier. If x_t is not an outlier, we directly output its clustering result. Otherwise, we use these outliers for subspace structure change detection. If the change is detected based on our proposed criteria, we update the subspace structure and the representative set via performing SSC on a small subset to replace the previous representative set. We repeat the "subspace structure change detection - subspace structure update" procedure until the end of the stream. Note that the subspace change detection is simplified here for illustration.

After initialization, for each new arriving point x_t, we decide whether x_t is an outlier based on its sparse linear combination over the current universal representative set. If x_t is not an outlier, we directly output its clustering result. Otherwise, we use these outliers for subspace structure change detection. If the change is detected based on our proposed criteria, we update the subspace structure and the representative set via performing SSC on a small subset to replace the previous ones. Our method repeats this "subspace structure change detection - subspace structure update" procedure until the end of the stream. The whole pipeline of our method is presented in Algorithm 1. In the rest of this paper, we omit the superscript t for simplicity when it does not cause misunderstanding from the context.

C. SELECT REPRESENTATIVES

In this subsection, we specify our strategy to select representatives for subspaces. Because we require the representatives of each subspace $\mathcal{S}_i$ to span it, the number of representatives n_i should not be smaller than subspace dimension d_i of $\mathcal{S}_i$. Intuitively, if we select more representatives from $\mathcal{S}_i$, we could get a better estimation of the structure of $\mathcal{S}_i$. However, as the size of representatives becomes larger, the update of subspaces during the later stage of the stream will become less efficient. This is because that SSC is sensitive to the number of points and the subsequent SSC problem to update subspace structure is related to the size of the current universal representative set. Therefore, we need an efficient sampling

Algorithm 1 Stream Sparse Subspace Clustering

```
Input: Stream $X = \{x_1, \cdots, x_t, \cdots\}$
Parameters: Initialization size $T_0$, Upper bound of
            subspace dimension $d$, Subspace change
            detection threshold $\gamma$
Output: Clustering result
$\beta \leftarrow 2d$;
$[\{C_i^{(T_0)}\}_{i=1}^{k_0}, k_0] \leftarrow \{x_t\}_{t=1}^{T_0}$;
$\mathcal{R}_i^{(T_0)} \leftarrow C_i^{(T_0)}$;
$\mathcal{R} \leftarrow \cup_i \mathcal{R}_i^{(T_0)}$;   $k \leftarrow k_0$;
$t \leftarrow T_0 + 1$;
repeat
    $\hat{\alpha} \leftarrow x_t, \mathcal{R}$ by Eq. 3;
    if $\|\hat{\alpha}\|_0 < \beta$ then        // $x_t$ is not outlier
        Output label of $x_t$ by Eq. 4;
        $t \leftarrow t + 1$;
    else        // Subspace change detection
        $m \leftarrow 0$;   $L \leftarrow kd$;
        for $j \leftarrow 1, \cdots, L$ do
            $\hat{\alpha} \leftarrow x_{t+j}, \mathcal{R}$ by Eq. 3;
            if $\|\hat{\alpha}\|_0 < \beta$ then        // $x_{t+j}$ is not outlier
                Output label of $x_{t+j}$ by Eq. 4;
            else $m \leftarrow m + 1$ ; // $x_{t+j}$ is outlier
        end
        if $m > \gamma$ then        // Subspaces update
            $[\{C_i^{(t+L)}\}_{i=1}^{k}, k] \leftarrow \mathcal{R} \cup \{x_{t+j}\}_{j=1}^{L}$;
            $\mathcal{R}_i^{(t+L)} \leftarrow C_i^{(t+L)}$;
            $\mathcal{R} \leftarrow \cup_i \mathcal{R}_i^{(t+L)}$;
        end
        $t \leftarrow t + L + 1$;
    end
until the end of the stream;
```

strategy to select representatives from each subspace so that the sampling complexity n_i could be as small as possible.

Our research reveals that to ensure $span(\mathcal{R}_i) = \mathcal{S}_i$, one only need to randomly sample d_i points from the corresponding cluster C_i, as long as points that lie on the subspace $\mathcal{S}_i$ are uniformly distributed. Therefore, the minimal requirement for n_i is exactly d_i, as stated in the following proposition.

Proposition 1: Given a subspace $\mathcal{S}$ with dimension d, suppose $r^1, \cdots, r^d \in \mathcal{S}$ are independently and identically distributed random points sampled from uniform distribution on $\mathbb{S}^{D-1} \cap \mathcal{S}$, where $\mathbb{S}^{D-1}$ is the unit sphere in $\mathbb{R}^D$, then with high probability, we have

$$span(r^1, \cdots, r^d) = \mathcal{S}.$$

Proof: Assume A is an orthonormal basis of $\mathcal{S}$, and $r^i = Ay^i$ for $i = 1, \cdots, d$, then the coefficients $y^1, \cdots, y^d \in \mathbb{R}^d$ are independently and identically distributed random points sampled uniformly from $\mathbb{S}^{d-1}$. Uniform distribution on $\mathbb{S}^{d-1}$ can be generated from standard Gaussian distribution [27]. Specifically, let $z \in \mathbb{R}^d$ be a random vector distributed as $\mathcal{N}(0, I_d)$, then $\hat{z} = z/\|z\|_2$ is uniformly distributed on $\mathbb{S}^{d-1}$. So it suffices to show that any family of independently and identically distributed random points $z_1, \cdots, z_d$ sampled from $\mathcal{N}(0, I_d)$ are linear independent with high probability, which is equivalent to bound the smallest singular value $\sigma_d(Z)$ of $Z = [z_1, \cdots, z_d]$ from 0. This has been proved [28] as follows,

$$P(\sqrt{d}\sigma_d(Z) \le t) = 1 - e^{-t^2/2-t} + O(d^{-c}),$$

where $c > 0$ is an absolute constant. □

Actually, one can compute the exact distribution of $\sigma_d(Z)$ explicitly [28], [29]. For a typical case in our synthetic evaluation where the dimension of subspace $d = 5$, we get $P(\sqrt{5}\sigma_5(Z) \le 0.001) \le 0.001$ by numerical computation. Then we conclude that a random sampling of representatives $r^1, \cdots, r^5$ from the corresponding cluster could guarantee $P(\text{rank}(r^1, \cdots, r^5) = 5) \ge 99.9\%$.

To further illustrate this, we generate a set S of 1000 unit norm points uniformly distributed on a subspace with dimension d in $\mathbb{R}^{500}$. We randomly select n points from S, then we compute the rank of the column space spanned by these selected points. Let n vary from 2 to 20 and d vary from 5 to 10, in each case, we repeat 100 times and take the average. The result is shown in Fig 2. We can see that for each d, only $n = d$ randomly selected points are linear independent, thus could serve as representatives.

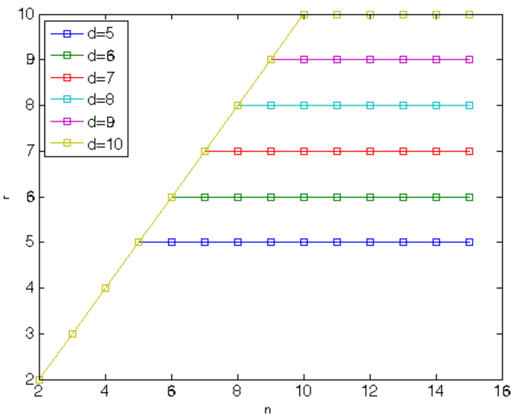


FIGURE 2. The rank of the randomly selected points from the subspace with respected to the sample size. The figure shows that for a subspace of dimension *d*, only *n* = *d* randomly selected points are enough to span the subspace. This verifies our claim in Proposition 1.

In practice, since both the dimension of subspace and distribution of points on subspace are usually unknown, one could choose n_i as a slightly larger estimation of d_i.

D. SUBSPACE STRUCTURE CHANGE DETECTION

Given a new arriving point x_t at time t, we express x_t as a sparse linear combination over the current universal representative set $\mathcal{R}$ with coefficient of the form $\alpha = [\alpha_1^T, \cdots, \alpha_k^T]^T$ and each $\alpha_i \in \mathbb{R}^{n_i}$, then we aim to solve the following unconstrained problem:

$$\min_{\alpha} \|x_t - \sum_i R_i \alpha_i\|_2^2 + \lambda \sum_i \|\alpha_i\|_1, \tag{3}$$

where $R_i = [r_i^1, \cdots, r_i^{n_i}] \in \mathbb{R}^{D \times n_i}$, $i = 1, \cdots, k$, and $\lambda > 0$ is the regularization parameter. Here we use the regularized version of SSC because data contains noise in practice.

From theoretical study in [30], if x_t lies in certain subspace $\mathcal{S}_{i^*}$, then the nonzero components in the optimal solution $\hat{\alpha} = [\hat{\alpha}_1^T, \cdots, \hat{\alpha}_k^T]^T$ of the problem (3) corresponding to representatives from other subspaces will be sufficiently small compared with components corresponding to representatives from the subspace $\mathcal{S}_{i^*}$. In this case, we simply output the clustering result of x_t determined by

$$i^* = \arg\max_i \|\hat{\alpha}_i\|_1. \tag{4}$$

In contrast, if x_t is an outlier, which means that x_t does not belong to any currently observed subspaces, the nonzero components in the optimal $\hat{\alpha}$ tend to spread over different subspaces, which results in a large l_0 norm of $\hat{\alpha}$. Therefore, we regard a point as an outlier if the number of nonzero components in the optimal solution $\hat{\alpha}$ is larger than a threshold β. We set $\beta = 2d$ in our algorithm.

A key observation is that the appearance of the outlier implies a potential change of subspace structure soon. In this case, we confirm further whether the change is really taking place. To achieve this goal, when encountered with the first outlier x_t, we start a temporary session to study the frequency of outliers appearing in the following points. More precisely, we take a session of points $W_L^t = \{x_{t+1}, \cdots, x_{t+L}\}$ of length L into consideration. If the number of outliers in the session W_L^t is larger than a threshold γ, these outliers probably come from some new subspaces that do not be reflected by the current representative set. Then it is time to update the subspace structure.

Otherwise, these outliers are not sufficient to affect the observed structure of subspaces, and they were judged as outliers possibly because of the noise involved in these points. As a result, we keep the subspace structure unchanged and go on clustering new points regularly as we did before until the next outlier appears.

E. SUBSPACE STRUCTURE UPDATE

To update the structure of subspaces, we simply perform SSC over the union $\mathcal{R} \cup W_L^t$ to segment these points into new subspaces. The motivation is as follows. First, the current universal representative set $\mathcal{R}$ can well describe the structure of observed subspaces; second, points in the session W_L^t incorporate potential change emerging in the stream.

The number of subspaces is also self-tuned by the method proposed in [26], as we did in initialization. We randomly select n_i points from each cluster C_i to act as the representative set for the corresponding subspace, as we have justified in Proposition 1. Then we use the selected representatives to replace the previous ones.

During our iterative update of subspaces, the representatives of all the subspaces that have ever appeared are stored in the universal representative set. Thus representatives also help to record the evolution of subspaces from the beginning to the end of the stream.

F. IMPLEMENTATION

In this subsection, we provide a more detailed explanation of the implementation of our algorithm. We introduce a parameter d in Algorithm 1. This is because it is difficult to estimate the size of the representative set n_i for each subspace $\mathcal{S}_i$. For convenience, we use an estimated uniform parameter d as an upper bound of d_i's. In this way, we only need to store at most $Kd + L$ points in memory. We set $L = Kd$ in practice, so the overall memory complexity is $O(Kd)$. Meanwhile, the size of the universal representative set is at most Kd, so the outlier detection steps by solving the problem can be executed extremely fast.

IV. EXPERIMENTS

A. EXPERIMENT SETUP

In this section, we compare the performance of our method with two stream subspace clustering methods on both synthetic and real-world streams. All the algorithms are implemented in MATLAB 2016b and run on a Ubuntu 16.04 server with Intel Xeon 3.0 GHz CPU, 64 GB memory.

1) COMPARED METHODS

We compare with the following two state-of-art stream subspace clustering methods: OLRSC [12] and ORPCA [23]. For both of them, we apply K-means to the obtained low-rank representations of points. This strategy is also employed in OLRSC [12]. We note that the number of clusters is changing in our setting, while OLRSC and ORPCA require a predefined number of clusters to perform K-means. For a fair comparison, we adopt the number of clusters obtained by our method during the data stream to serve as the number of clusters for OLRSC and ORPCA. In both compared algorithms, we use the codes released by the authors and set parameters as suggested in the corresponding paper [12], [23].

2) EVALUATION METRIC

Given a stream $X = \{x_1, \cdots, x_N\}$, we record the time indices $\{T_s\}_{s=1}^S$ of StreamSSC to update subspaces, where S is the number of updates. That is, change in the subspace structure is detected by StreamSSC and thus SSC is invoked at the T_s-th point, $s = 1, \cdots, S$. Then we divide the $N - T_0$ stream points (the T_0 points for initialization excluded) into $S+1$ successive sessions $\{G_s\}_{s=1}^{S+1}$ with each $G_s = \{x_{T_{s-1}+1}, \cdots, x_{T_s}\}$ for $1 \leq s \leq S$ and $G_{S+1} = \{x_{T_S+1}, \cdots, x_N\}$, such that $N - T_0 = \sum_{s=1}^{S+1} |G_s|$. Denote P_s as ground truth labels of points in G_s and Q_s as the corresponding clustering results, then $\text{NMI}(P_s, Q_s)$ is the normalized mutual information of clustering results for G_s. We use two measurements to evaluate the clustering performance: mean normalized mutual information (mNMI) and weighted normalized mutual information (wNMI), defined as follows:

$$\text{mNMI} = \frac{1}{S+1}\sum_{s=1}^{S+1}\text{NMI}(P_s, Q_s), \tag{5}$$

$$\text{wNMI} = \sum_{s=1}^{S+1}\frac{|G_s|}{\sum_{s=1}^{S+1}|G_s|}\text{NMI}(P_s, Q_s), \tag{6}$$

We adopt wNMI as a measurement because the number of points in session $|G_s|$ may vary for different s. Both mNMI and wNMI range from 0 to 1, and a higher value means a better clustering result.

3) STREAM DESIGN

For a dataset $X = \{x_1, \cdots, x_N\}$ containing K clusters $\{C_i\}_{i=1}^{K}$, we partition each cluster C_i into five subgroups $\{C_i^j\}_{j=1}^{5}$ of (approximately) equal size. The entire stream is composed of K successive sessions $\{X_i\}_{i=1}^{K}$. Each session X_i contains five subgroups belonging to disjoint clusters. To assess the effect of order among points on the performance of clustering, all points within one session are randomly permuted. In this case, we can imitate the following three kinds of change in the structure of subspaces: (1) disappearing cluster (e.g. subspace "1" disappears from X_1 to X_2), (2) emerging cluster (e.g. subspace "6" emerges from X_1 to X_2), (3) reappearing cluster (e.g. subspace "1" reappears from X_6 to X_7). An example of the stream for $K = 10$ is illustrated in Table 1.

TABLE 1. Stream design. The stream X is composed of 10 successive sessions $\{X_i\}_{i=1}^{10}$. Each session X_i consists of five subgroups belonging to disjoint clusters. The subgroups included in each X_i are shown in the same column below X_i. For example, the first session X_1 consists points of clusters with labeling 1, 2, 3, 4, and 5.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
C_1^1	C_2^2	C_3^3	C_4^4	C_5^5	C_6^5	C_7^5	C_8^5	C_9^5	C_{10}^5
C_2^1	C_3^2	C_4^3	C_5^4	C_6^4	C_7^4	C_8^4	C_1^4	C_{10}^4	C_1^5
C_3^1	C_4^2	C_5^3	C_6^3	C_7^3	C_8^3	C_9^3	C_{10}^3	C_1^4	C_2^5
C_4^1	C_5^2	C_6^2	C_7^2	C_8^2	C_9^2	C_{10}^2	C_1^3	C_2^4	C_3^5
C_5^1	C_6^1	C_7^1	C_8^1	C_9^1	C_{10}^1	C_1^2	C_2^3	C_3^4	C_4^5

All the experiments are repeated 10 times with different random seeds and the average results are reported.

B. STREAM DATA

1) SYNTHETIC STREAM

We generate $K = 10$ subspaces $\{\mathcal{S}_i\}_{i=1}^{10}$ in $\mathbb{R}^{1000}$. The number of points in each subspace is n and the total number of points is $N = K * n$. Each $\mathcal{S}_i$ is of dimension d with default value 5, and spanned by a linear combination of d basis vectors. Both the components of the basis vectors and the coefficients of linear combination of a clean data points $\bar{x}_t$ over basis vectors are uniformly sampled from [0, 1]. Denote the matrix of clean points as $\bar{X} = [\bar{x}_1, \cdots, \bar{x}_N] \in \mathbb{R}^{1000\times N}$, where each point $\bar{x}_t$ is normalized to have unit norm. To test robustness of the methods, we introduce noise in the generated data. Let μ be the average of all the entries in $\bar{X}$, then we generate noisy point x_t from $\bar{x}_t$ such that the j-th component $x_t(j) = \bar{x}_t(j) + \rho\mu\sigma$ for each $t = 1, \cdots, N$ and $j = 1, \cdots, 1000$, where $\sigma \sim \mathcal{N}(0, 1)$ is the standard Gaussian variable and ρ is the noise scale with default value 0.3.

2) REAL WORLD STREAM

We evaluate on a stream produced by the public dataset: MNIST (handwritten digit images). We choose $n = 1000$ images from each class of training set of MNIST. Each image is represented as a $D = 784$ dimensional vector of unit norm. The stream consists of $N = 10,000$ images and is segmented into 10 successive sessions as shown in Table 1. To test the scalability of our method, we also use the full training set of MNIST, which consists of $n = 6,000$ images from each class. Then the stream size is $N = 60,000$. The stream design is the same as what we do for $N = 10,000$ images. Evaluation of scalability will be shown in Table 7.

C. RESULT ON SYNTHETIC STREAM

1) SUBSPACE CHANGE DETECTION

First, we show the capability of StreamSSC to detect the change in the subspace structure. A typical case of stream size $N = 10000$, subspace dimension $d = 5$, noise scale $\rho = 0.3$ is shown in Table 2. We can see that StreamSSC performs five subspace updates in total during the stream. The detected subspace numbers keep consistent with the ground truth increase in the number of subspaces. We also note that the detected subspace structure change time indices are very close to the actual change time indices. For instance, StreamSSC detects subspace structure change at time $T_2 = 2082$, following the actual change time $t = 2000$, the beginning of the session X_3. The subspace structure changes at this time because points in the cluster C_7 start to arrive, i.e., the subgroup C_7^1 shown in Table 1 appears. This implies that our method can effectively detect changes in the subspace structure during the stream. Note that StreamSSC does not update subspaces after $T_5 = 5070$ anymore. This is obvious since all the 10 subspaces have been detected and the current universal representative set has already contained representatives from all the 10 subspaces.

TABLE 2. Illustration of subspace structure change detection by StreamSSC on a synthetic stream of size $N = 10,000$, subspace dimension $d = 5$, noise scale $\rho = 0.3$.

s	1	2	3	4	5
Actual change time index	1000	2000	3000	4000	5000
Detected change time index	1076	2082	3058	4063	5070
Detected subspace number	6	7	8	9	10

TABLE 3. Performance of StreamSSC, OLRSC and ORPCA w.r.t. noise scale ρ on synthetic stream. The stream contains $N = 10000$ points lying in $K = 10$ subspaces. The subspace dimension $d = 5$ and noise scale ρ varies from 0.1 to 0.5 with step 0.1.

ρ	mNMI (%)			wNMI (%)			run time(sec.)		
	StreamSSC	OLRSC	ORPCA	StreamSSC	OLRSC	ORPCA	StreamSSC	OLRSC	ORPCA
0.1	**92.60±1.58**	88.98±1.33	87.55±1.36	**95.01±1.10**	92.07±0.93	88.65±4.05	**0.94**	10.28	9.88
0.2	**94.47±0.76**	91.98±1.64	89.89±1.05	**96.20±1.49**	92.74±1.92	90.21±1.60	**1.02**	10.40	10.62
0.3	**94.36±0.71**	92.55±0.86	90.12±1.98	**96.55±0.32**	93.77±1.06	90.98±2.31	**1.12**	10.79	10.93
0.4	90.94±1.91	**91.41±1.75**	87.99±2.43	**92.75±1.70**	92.51±1.33	88.41±3.35	**0.77**	10.75	11.05
0.5	**91.08±2.86**	90.94±1.52	89.52±1.68	**92.50±1.02**	91.41±1.10	90.48±2.36	**0.84**	10.86	11.00

TABLE 4. Performance of StreamSSC, OLRSC and ORPCA w.r.t. subspace dimension d on synthetic stream. The stream contains $N = 10000$ points lying in $K = 10$ subspaces. The noise scale $\rho = 0.3$ and subspace dimension d varies from 2 to 10.

d	mNMI (%)			wNMI (%)			run time(sec.)		
	StreamSSC	OLRSC	ORPCA	StreamSSC	OLRSC	ORPCA	StreamSSC	OLRSC	ORPCA
2	**91.60±3.64**	88.49±1.81	88.90±1.31	**92.52±4.93**	88.67±2.80	89.40±1.05	**0.70**	5.14	4.60
3	**95.79±0.74**	90.15±2.07	88.62±2.19	**96.50±1.07**	91.13±2.92	88.45±2.86	**0.85**	6.97	6.59
4	**95.09±0.99**	91.49±1.86	89.24±1.80	**96.86±1.19**	92.50±2.43	89.89±1.54	**1.04**	8.75	8.57
5	**94.36±0.71**	92.55±0.86	90.12±1.98	**96.55±0.32**	93.77±1.06	90.98±2.31	**1.14**	10.67	10.80
6	**93.42±0.79**	92.68±0.98	88.72±2.11	**96.17±0.40**	93.83±1.52	89.77±2.10	**1.19**	13.38	13.07
7	**93.61±0.48**	92.57±1.33	90.51±1.89	**96.08±0.24**	93.95±1.43	91.86±2.38	**1.22**	15.60	16.03
8	**92.68±1.00**	92.19±0.84	89.25±2.13	**94.71±1.55**	92.71±1.31	90.30±2.09	**1.04**	17.33	18.96
9	**92.86±1.09**	90.72±1.84	90.46±2.56	**95.27±1.04**	92.58±1.85	91.82±2.06	**1.17**	19.62	21.40
10	**91.55±0.81**	90.28±1.62	89.79±4.17	**94.67±0.65**	91.84±1.67	90.36±2.49	**1.31**	22.63	24.92

2) COMPARISON OF CLUSTERING PERFORMANCE

To evaluate the performance of StreamSSC, OLRSC, and ORPCA, we run on the synthetic stream of size $N = 10000$. First, we fix subspace dimension $d = 5$ and let noise scale ρ vary between $0.1 \sim 0.5$ with step 0.1, then we fix $\rho = 0.3$ and let d vary between $2 \sim 10$. Based on our conclusion from Proposition 1, the number of representatives selected from each subspace should be slightly larger than d to guarantee the clustering performance, we set it as $d + 1$. And the threshold γ used to detect the subspace structure change is set 3 as a default value. The results are reported in Table 3 and Table 4.

From both tables, we observe that StreamSSC outperforms OLRSC and ORPCA in clustering accuracy. Considering that OLRSC and ORPCA use the number of clusters produced by StreamSSC to perform K-means, and both the two measurements mNMI and wNMI range from $0.85 \sim 0.95$, we conclude that StreamSSC could capture the change of subspace structure rather accurately. In terms of running time, StreamSSC is about a magnitude faster than OLRSC and ORPCA. The results also verify the effectiveness of our proposed representative selection strategy and sampling complexity. Furthermore, we find that StreamSSC is not sensitive to the order of points in the stream.

3) SCALABILITY ANALYSIS

To compare the scalability of StreamSSC, OLRSC and ORPCA, we evaluate on seven synthetic streams of different sizes: $N = \{1 \times 10^4, 1 \times 10^5, 1 \times 10^6, 1 \times 10^7, 1 \times 10^8\}$. The time cost (seconds) is reported in Table 5. Although computational times of all the three methods grow almost linearly with the size of the stream, StreamSSC is about a magnitude faster than OLRSC and ORPCA. The reason is that to update the representative set in StreamSSC, we only need to perform SSC on a subset of at most $2K(d+1) = 120$ points, and the clustering process for a new arriving point to solve Eq. 3 is extremely fast. In contrast, OLRSC and ORPCA need to compute the low dimensional representation and update the dictionary together for every new arriving point, which requires more computation.

D. RESULT ON REAL WORLD STREAM

We set the rank of the dictionary used in OLRSC and ORPCA as $5K = 50$, inconsistent with the OLRSC paper [12], and the estimated dimension of each subspace as $d = 10$ for StreamSSC. The results are reported in Table 6.

We observe that StreamSSC outperforms OLRSC and ORPCA more than 5% on average. Regarding efficiency, the running time of StreamSSC is less than 20% of the time cost by OLRSC and ORPCA. Meanwhile, it is important to note that StreamSSC can output real-time clustering results for each new arriving point, while OLRSC and ORPCA could not report the clustering results until all the points in the stream are received.

For scalability, the results are reported in Table 7. We evaluate the running time on two cases: $N = 10,000$ and $N = 60,000$. We can see that SteamSSC scales well for real-world datasets. The running time is almost linear will the size of

TABLE 5. Comparison of scalability of three methods on five synthetic streams of different sizes: $N = \{1 \times 10^4, 1 \times 10^5, 1 \times 10^6, 1 \times 10^7, 1 \times 10^8\}$. Time cost(seconds) is reported. It shows that StreamSSC is about a magnitude faster than both OLRSC and ORPCA.

Methods	1×10^4	1×10^5	1×10^6	1×10^7	1×10^8
OLRSC	10.9	103	1.04×10^3	1.05×10^4	1.09×10^5
ORPCA	10.9	108	1.03×10^3	1.03×10^4	1.07×10^5
StreamSSC	1.1	10.1	104	1.07×10^3	1.12×10^4

TABLE 6. Performance of StreamSSC, OLRSC and ORPCA on a stream of MNIST dataset with stream size 10,000.

Methods	mNMI (%)	wNMI (%)	run time (sec.)
StreamSSC	**66.97± 4.61**	**67.04±4.59**	**6.2**
OLRSC	61.31± 3.67	61.24±3.64	43.3
ORPCA	62.26 ± 3.61	62.17±3.60	33.7

TABLE 7. Comparison of scalability of StreamSSC, OLRSC and ORPCA on streams of the MNIST dataset with stream size 10,000 and 60,000, respectively.

N	Methods	run time (sec.)
10,000	StreamSSC	**6.2**
	OLRSC	43.3
	ORPCA	33.7
60,000	StreamSSC	**41.2**
	OLRSC	260.9
	ORPCA	217.7

the stream. The growth rate is consistent with what we have observed for synthetic data.

V. CONCLUSION

In this paper, we propose a novel method of sparse subspace clustering for stream data (StreamSSC). StreamSSC can detect changes in subspace structure and feedback real-time clustering results of each point during the stream. Technically, we exploit the representative set to characterize the observed subspaces, then changes in subspace structure are captured by an outlier detection scheme. Our method is efficient in both computation and memory. Evaluation results on both synthetic and real-world streams demonstrate that our method outperforms the state-of-art stream subspace clustering methods.

REFERENCES

[1] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 2790–2797.

[2] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 663–670.

[3] S. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 10, pp. 1832–1845, Oct. 2010.

[4] S. Li, K. Li, and Y. Fu, "Temporal subspace clustering for human motion segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4453–4461.

[5] M. Soltanolkotabi and E. J. Candès, "A geometric analysis of subspace clustering with outliers," *Ann. Statist.*, vol. 40, no. 4, pp. 2195–2238, Aug. 2012.

[6] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013.

[7] S. Wang, B. Tu, C. Xu, and Z. Zhang, "Exact subspace clustering in linear time," in *Proc. AAAI*, 2014, pp. 2113–2120.

[8] C. You, D. P. Robinson, and R. Vidal, "Scalable sparse subspace clustering by orthogonal matching pursuit," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3918–3927.

[9] Y. Zhang, M. Zhao, and W. Kong, "Scalable subspace clustering via adaptive low-rank model," in *Proc. IEEE Symp. Product Compliance Eng.*, Dec. 2018, pp. 1–5.

[10] Y. Chen, C.-G. Li, and C. You, "Stochastic sparse subspace clustering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4155–4164.

[11] S. Matsushima and M. Brbic, "Selective sampling-based scalable sparse subspace clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 12416–12425.

[12] J. Shen, P. Li, and H. Xu, "Online low-rank subspace clustering by basis dictionary pursuit," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 622–631.

[13] Y.-X. Wang, H. Xu, and C. Leng, "Provable subspace clustering: When lrr meets ssc," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 64–72.

[14] P. Favaro, R. Vidal, and A. Ravichandran, "A closed form solution to robust subspace estimation and clustering," in *Proc. CVPR*, Jun. 2011, pp. 1801–1807.

[15] R. Vidal and P. Favaro, "Low rank subspace clustering (LRSC)," *Pattern Recognit. Lett.*, vol. 43, pp. 47–61, Jul. 2014.

[16] G. Liu and S. Yan, "Latent low-rank representation for subspace segmentation and feature extraction," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1615–1622.

[17] C.-G. Li and R. Vidal, "Structured sparse subspace clustering: A unified optimization framework," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, May 2015, pp. 277–286.

[18] X. Peng, L. Zhang, and Z. Yi, "Scalable sparse subspace clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 430–437.

[19] C. Yang, D. Robinson, and R. Vidal, "Sparse subspace clustering with missing entries," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2463–2472.

[20] Y. Wang, Y.-X. Wang, and A. Singh, "A deterministic analysis of noisy sparse subspace clustering for dimensionality-reduced data," in *Proc. ICML*, 2015, pp. 1422–1431.

[21] B. Nasihatkon and R. Hartley, "Graph connectivity in sparse subspace clustering," in *Proc. CVPR*, Jun. 2011, pp. 2137–2144.

[22] P. A. Traganitis and G. B. Giannakis, "Sketched subspace clustering," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1663–1675, Apr. 2018.

[23] J. Feng, H. Xu, and S. Yan, "Online robust pca via stochastic optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 404–412.

[24] P. Li, J. Feng, X. Jin, L. Zhang, X. Xu, and S. Yan, "Online robust low-rank tensor modeling for streaming data analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1061–1075, Apr. 2019.

[25] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. De Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, p. 13, 2013.

[26] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Proc. NIPS*, vol. 17, 2004, pp. 1601–1608.

[27] R. Vershynin, *High-Dimensional Probability: An Introduction With Applications in Data Science*, vol. 47. Cambridge, U.K.: Cambridge Univ. Press, 2018.

[28] T. Tao and V. Vu, "Random matrices: The distribution of the smallest singular values," *Geometric Funct. Anal.*, vol. 20, no. 1, pp. 260–297, Jun. 2010.

[29] A. Edelman, "The distribution and moments of the smallest eigenvalue of a random matrix of wishart type," *Linear Algebra its Appl.*, vol. 159, pp. 55–80, Dec. 1991.

[30] E. Elhamifar, M. Soltanolkotabi, and S. Sastry, "Approximate subspace-sparse recovery with corrupted data via constrained ℓ_1-minimization," 2014, *arXiv:1412.7260*. [Online]. Available: http://arxiv.org/abs/1412.7260

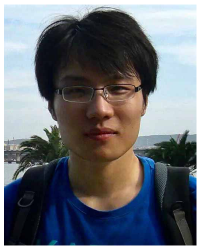
KEN CHEN is currently pursuing the Ph.D. degree. He is currently a Senior Engineer and a Leader of Digital Transportation with Sichuan Railway Industry Investment Group Company Ltd., an Expert with the Department of Science and Technology of Sichuan Province, China, and an honored Researcher with the Future Transportation Laboratory of Gaode Map. He has published more than 30 academic articles, contributed in five standardizations and two books, and holds ten patents. His research interests include intelligent transportation, space information technology, and AI and big data applications. He was named an Outstanding Contributor of Informatization by Sichuan Provincial Entrepreneurs' Association, in 2020.

YONG TANG is currently a Professorial Senior Engineer, the Secretary of the Party Committee, the Chairman of the Board, the CEO of Sichuan Railway Industry Investment Group Company Ltd., and a leader of more than ten provincial and ministerial key research projects. His research interests include big data in transportation, intelligent transportation systems (ITS), and highway mechanical and electrical systems. He has won several honored awards, including the Distinguished Management Innovation Leader by the State-Owned Assets Supervision and Administration Commission (SASAC) and the Outstanding Youth Entrepreneur of Sichuan Province by the Sichuan Provincial Youth Entrepreneurs' Association.

LONG WEI received the Ph.D. degree from Zhejiang University. He is currently an Algorithm Engineer II with the Alibaba DAMO Academy. His research interests include machine learning and computer vision.

PENGFEI WANG received the Ph.D. degree from the Chinese Academy of Sciences. He is currently an Algorithm Engineer II with the Alibaba DAMO Academy. His research interests include spatiotemporal data mining and visual intelligence.

YONG LIU received the master's degree from Sichuan University, in 2011. He was a Video Cloud Expert with Huawei and a Senior Engineer with Cisco. He is currently a Senior Solutions Architect with the Alibaba Cloud. He is focusing on machine learning and the IoT in transport field.



ZHONGMING JIN received the Ph.D. degree from Zhejiang University, in 2015. He was a Researcher with the Baidu Research, Beijing, China. He is currently a Staff Algorithm Engineer with the Alibaba DAMO Academy. His research interests include large scale machine learning and computer vision.