IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 5, 2022, accepted June 3, 2022, date of publication June 13, 2022, date of current version June 17, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3182345

Kernel Clustering With Sigmoid Regularization for Efficient Segmentation of Sequential Data

TUNG DOAN[1] AND ATSUHIRO TAKASU[2,3], (Member, IEEE)
[1]School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi 11615, Vietnam
[2]Graduate University for Advanced Studies, SOKENDAI, Hayama, Kanagawa 240-0193, Japan
[3]National Institute of Informatics, Tokyo 101-8430, Japan

Corresponding author: Tung Doan (tungdp@soict.hust.edu.vn)

ABSTRACT The segmentation of sequential data can be formulated as a clustering problem, where the data samples are grouped into non-overlapping clusters with the constraint that all members of each cluster are in a successive order. A popular algorithm for optimally solving this problem is dynamic programming (DP), which has quadratic computation and memory requirements. Given that sequences in practice are too long, this algorithm is not a practical approach. Although many heuristic algorithms have been proposed to approximate the optimal segmentation, they have no guarantee on the quality of their solutions. In this paper, we take a differentiable approach to alleviate the aforementioned issues. First, we introduce a novel sigmoid-based regularization to smoothly approximate the constraints. Combining it with objective of the balanced kernel clustering, we formulate a differentiable model termed *Kernel clustering with sigmoid-based regularization* (KCSR), where the gradient-based algorithm can be exploited to obtain the optimal segmentation. Second, we develop a stochastic variant of the proposed model. By using the stochastic gradient descent algorithm, which has much lower time and space complexities, for optimization, the second model can perform segmentation on overlong data sequences. Finally, for simultaneously segmenting multiple data sequences, we slightly modify the sigmoid-based regularization to further introduce an extended variant of the proposed model. Through extensive experiments on various types of data sequences performances of our models are evaluated and compared with those of the existing methods. The experimental results validate advantages of the proposed models. Our Matlab source code is available on github.

INDEX TERMS Sequence segmentation, sequential data, differentiable approximation, stochastic optimization, change point detection, temporal clustering.

I. INTRODUCTION

Recently, there has been an increasing interest in developing machine learning and data mining methods for sequential data. This is due to the exponential growing in number of collected data sequences from applications in wide range of fields, including computer vision [1]–[3], speech processing [4]–[6], finance [7]–[9], bio-informatics [10], [11], climatology [12]–[15] and traffic monitoring [16]–[18]. The main problem associated with analysis of these sequences is that they consists of a huge number of data samples. Therefore, it is desirable to summarize the whole sequences by a much smaller number of the data representatives, alleviating burden for the subsequent tasks.

Such compressed and concise summarization can be obtained via sequence segmentation. More specifically, this aims at partitioning the data sequences into several non-overlapping and homogeneous segments of variable durations, in which some characteristics remain approximately constant. It is widely recognized in the literature that the segmentation of sequential data can be considered as a clustering problem. The difference is that all data samples of each cluster, which represents a segment, are constrained to be in a successive order. Thus, in this paper, we focus on clustering-based methods for segmentation of data sequences.

In practice, sequential data are often composed of nonlinear and complex segments. Therefore, kernel methods are often applied to map data samples into a new feature space before segmenting. Due to the constraint imposed on the data samples in each cluster, traditional algorithms for clustering are inapplicable to the segmentation problem. [19] proposed

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

an optimal algorithm based on dynamic programming (DP) for segmenting data sequence in the features space, which is associated with a pre-specified kernel and mapping functions. In general, DP has quadratic time and memory complexities. It even induces running time of order $O(n^4)$,[1] where n is the length of the sequence, in practice. Therefore, it is intractable to perform segmentation on long data sequence using DP-based algorithms. To alleviate this issue, many attempts have been made to create approximations to the optimal algorithm. Although a considerable amount of the computational costs are reduced, there are still critical drawbacks remained in the approximation algorithms. Taking pruned DP [20] and greedy algorithm [21] as representatives. These methods sequentially partition the data sequence, returning one segment boundary (*a.k.a*, change point) at each iteration. This strategy offers a reduction in the computational time. However, its expense is that errors might occur at the earlier steps and they would influence on the subsequent iterations, inducing a huge bias in the final results. Massive memory complexity is also a vital drawback of almost kernel-based methods. They need store the kernel matrix, which requires order of $O(n^2)$ space. Therefore, they are prohibited by themselves from handling extensively long data sequences.

In this paper, we take a different approach to alleviate the aforementioned issues. More precisely, we introduce a novel sigmoid-based regularization, which smoothly approximates the constraints of the segmentation problem. It is then integrated with balanced kernel clustering to perform segmentation on sequential data. Our method owns several preferable characteristics. First, because objective of the proposed model is differentiable w.r.t unconstrained and continuous variables we can easily optimize it using gradient descent GD algorithm. Different from the existing methods, which are just heuristic approximations of the optimal segmentation algorithm, our model has a guarantee on quality of the solutions as convergence of the GD algorithm was theoretically proved [22]. Second, the proposed model offers the applicability of a more efficient optimization algorithm based on stochastic gradient – the gradient that is estimated from a subsequence (mini-batch), which is randomly sampled from the original data sequence at each iteration. Therefore, the stochastic variant of our model has much lower time and space complexities, making segmentation of extensively long data sequences possible. Finally, the proposed model is flexible. We can easily modify the sigmoid-based regularization to further form a new extended variant that can simultaneously segment multiple data sequences. Through extensive experiments on various types of sequential data, our models are evaluated and compared with baseline methods. The results validate advantages of the proposed models. In summary, contributions of this paper are as follows

- Introduction of sigmoid-based regularization that enables kernel clustering to partition sequential data. Objective of the proposed method called *Kernel clustering with sigmoid regularization* (KCSR) is smooth and can be effectively solved using gradient-based algorithm.
- Development of a stochastic variant of KCSR to reduce the memory complexity, which is prominent in almost kernel-based methods that prohibits them from handling large-scale datasets.
- Extension of KCSR for simultaneously segmentation of multiple data sequences.
- Extensively empirical evaluation of the proposed methods on widely public datasets shows theirs superiorities over the existing methods.

The rest of this paper is organized as follows: In Section II, we review related works that perform segmentation based on clustering methods. Next, we briefly presents some background for our proposed models in Section III. Section IV introduces the proposed model KCSR and its stochastic version. This section also describes how to modify the sigmoid-based regularization to form an extension of KCSR that can simultaneously segment multiple data sequences. After illustrating and discussing experimental results in Section V, we conclude the paper in Section VI.

II. RELATED WORKS

In this paper, we focus on clustering-based methods for nonlinear segmentation of sequential data. Thus, we will review related works in the literature of kernel segmentation, which sometime is referred to as *offline kernel change point detection (CPD)* [23]. Here, the change points indicate the boundaries between the segments. In addition, we also review *temporal clustering* methods. They have recently gained more and more popularity in the computer vision field, where clustering-based algorithms are employed to segment videos of human motions.

Offline Kernel Change Point Detection: According to [23], almost all offline kernel CPD methods attempt to optimize the objective function as defined in (2). This is also the objective of the kernel k-means clustering. Based on the search scheme for the segment boundaries, existing methods can be divided into local group, which uses sliding window and global group, which bases on dynamic programming.

The local methods [4], [24]–[27] slide a window with a large enough width over the data sequence. They then detect, in the window, a single change point, at which the difference between the preceding and succeeding samples is maximal. Although having low computational cost, these methods is sub-optimal as the whole sequence is not considered when detecting the changes. Our approach is more similar to the global methods, which take all data samples into account for change detection. [19], [28] employed dynamic programming (DP) algorithm to optimally obtain the segment boundaries. However, because DP have time complexity of order $O(n^4)$ (including computational time of the cost matrix [20] in the feature space), it is impractical for handling long data sequences. To reduce the time complexity, [21] proposed a greedy algorithm that sequentially detects change points

[1] including time for computing the cost matrix in the feature space [20]

one at an iteration. [20] further reduce the space requirement by introducing pruned DP, which combines low-rank approximation of the kernel matrix and binary segmentation algorithm. Our approach is different from these two methods as it searches for all the segment boundaries simultaneously. In addition, quality of its solutions is guaranteed as convergence to optimum of the gradient descent algorithm employed in our model is theoretically proved [22]. Both pruned DP and the greedy algorithm are heuristic approximations of the original DP. Since sequentially detect the changes, errors at the early iterations are propagated and can not be corrected at the subsequent iterations.

Temporal clustering refers to the factorization of data sequences into a set of non-overlapping segments, each of which belongs to one of k clusters. Maximum margin temporal clustering (MMTC) [29] and Aligned clustering analysis (ACA) [30] divide data sequences into a set of non-overlapping short segments. These subsequences are then partitioned into k classes using unsupervised support vector machine [29] or kernel k-means clustering [30]. Recently, a branch of methods based on subspace clustering has been proposed. These methods often include two steps. First, given a data sequences $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, they learn a new representation (coding matrix) $\boldsymbol{Z} = [z_1, \ldots, z_n]$ that characterizes the underlying subspaces structures and sequential (a.k.a. temporal) information of the original data. Second, the normalized cut algorithm (Ncut) [31] is then utilized for segmentation of $\boldsymbol{Z}$.

To preserve the sequential information in the new representation, [32], [33] proposed a linear regularization of the form $||\boldsymbol{ZR}||_{1,2}$, where $\boldsymbol{R} \in \mathbb{R}^{n\times(n-1)}$ is a lower triangular matrix with -1 on the diagonal and 1 on the second diagonal. By minimizing this regularization jointly with the subspace learning objective, the new representation $\boldsymbol{z}_j$ and $\boldsymbol{z}_{j+1}$ of the two consecutive samples $\boldsymbol{x}_j$ and $\boldsymbol{x}_{j+1}$, respectively, are forced to be similar. [1] further integrated a weight matrix into the linear regularization to avoid equally constraining on every pair of consecutive samples. Nevertheless, since the regularization is linear, it is ineffective for handling complex data structure. To leverage this issue, [34], [35] proposed manifold-based regularization that preserves the sequential information for the local neighborhood data samples. This type of regularization is more preferable [2] as it often outperforms the linear one in most tests [36]. Our approach also employs regularization to model sequential characteristics of the data. However, the sequential information is both globally and locally preserved in the proposed methods, thanks to the smoothness of the sigmoid functions. In addition, since the temporal regularization makes representation of consecutive samples become similar, boundaries of the segments become difficult to be identified. Our methods, in contrast, approximate the boundaries by midpoints in the summation of sigmoid functions with high steepness. Therefore, our models are expected to obtain better segmentation accuracy.

Both temporal clustering and offline kernel CPD approaches have to store an affinity graph matrix and/or a kernel matrix, which require memory of order $O(n^2)$. This is also a vital reason that inhibits them from handling long data sequence. Stochastic variant of our method has significantly lower space requirement. At each iteration, it approximates the gradient based on a partial kernel matrix, which corresponds to data samples in the current minibatch. Therefore, memory complexity of Stochastic KCSR is only $O(b^2)$, where $b \ll n$ is the minibatch size. Among the existing methods, only pruned DP in [20] is capable of handling large-scale data because it employs low-rank approximation of the kernel matrix, which only requires space of order $O(r^2)$, where $r \ll n$ is the rank of the approximation. Comparison between performances of Stochastic KCSR and this algorithm on large datasets will be given in Section V.

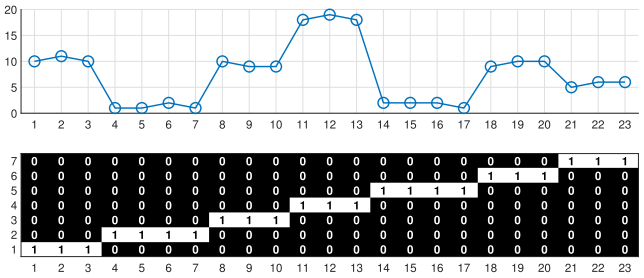


FIGURE 1. An example of sequence segmentation: (top) an example sequence of length 23 and (bottom) the corresponding indicator matrix with number of segments $k = 7$.

III. NOTATIONS AND BACKGROUND

A. NOTATIONS

Throughout this paper, we denote vectors and matrices by bold lower-case and bold uppercase letters, respectively. For a particular matrix $\boldsymbol{A}$, its i^{th} column is denoted as $\boldsymbol{a}_i$ and its element at position (j, i) is expressed by $a_{j,i}$ or $A_{j,i}$. The transpose matrix of $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^\top$. If $\boldsymbol{A}$ is a square matrix of size n then its trace is expressed as $\text{Tr}(\boldsymbol{A}) = \sum_{i=1}^{n} A_{i,i}$. If $\boldsymbol{A} \in \{0, 1\}^{k\times n}$ then for any given element $A_{j,i}$ we have $A_{j,i} = 0$ or $A_{j,i} = 1$ ($\boldsymbol{A}$ is a binary matrix). By $a \ll b$, we mean that a is very small in comparison with b.

B. KERNEL SEGMENTATION

The goal of the segmentation task is to partition a data sequence into several non-overlapping and homogeneous segments of variable durations. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ denotes the given sequence of length n and dimension d. For the number of segments k that is specified in advance, a valid solution of the $k-$segmentation problem can be represented by an sample-to-segment indicator matrix $\boldsymbol{G} \in \{0, 1\}^{k\times n}$, whose each element is as follows

$$G_{i,j} = \begin{cases} 1 & \boldsymbol{x}_j \in \text{segment } i, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

$\boldsymbol{G}$ must satisfy two constraints, including i) *Boundary:* $G_{1,1} = 1$ and $G_{k,n} = 1$ and ii) *Monotonicity:* for any given

$G_{i,j} = 1$ then for the next column $G_{i,j+1} = 1$ or $G_{i+1,j+1} = 1$. An example of the indicator matrix is given in Figure 1.

To discover segments with complex and nonlinear structures, kernelization is often applied. More specifically, the data sequence $\boldsymbol{X}$ is mapped onto some high dimensional space (*a.k.a.* feature space) associated with a pre-specified kernel function $\kappa(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. The mapping function $\phi(\cdot)$ is implicitly defined by $\phi(\boldsymbol{x}_i) = \kappa(\boldsymbol{x}_i, \cdot)$, resulting the inner-product $\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. A common objective for segmentation is to minimize the total summation of the intra-segment variances [23]. Thus, the optimization problem is often formulated as follows

$$\underset{\boldsymbol{G}\in\mathcal{G}}{\operatorname{argmin}} \sum_{j=1}^{k}\sum_{i=1}^{n} G_{j,i}\,||\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_j||_2^2, \tag{2}$$

where $\mathcal{G}$ is the set of all valid sample-to-segment indicator matrices and $\boldsymbol{\mu}_j$ is the mean of the j^{th} segment in the feature space. We can observe that the objective of this problem is similar to that of the kernel $k-$means and it is difficult to be minimized because $\boldsymbol{G}$ is the discrete variables with combinatorial constraints.

C. BALANCED KERNEL $k-$MEANS

As mentioned above, kernel segmentation is closely related to kernel $k-$means due to the similarity between their objectives. In fact, this objective can be rewritten in matrix form. More specifically, we can compute the corresponding kernel matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$, where each element $K_{i,j} = \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ represents how likely the two samples are assigned to the same class. Let $\boldsymbol{G} \in \{0,1\}^{k\times n}$ denotes the associated (unknown) sample-to-class indicator matrix of $\boldsymbol{X}$, where $G_{i,j} = 1$ if $\boldsymbol{x}_j$ is assigned to the i^{th} class and zero otherwise. Here, different from the segmentation task, there is no constraint on the indicator matrix $\boldsymbol{G}$. Then the objective function of kernel $k-$means [37]–[39] can be expressed as follows:

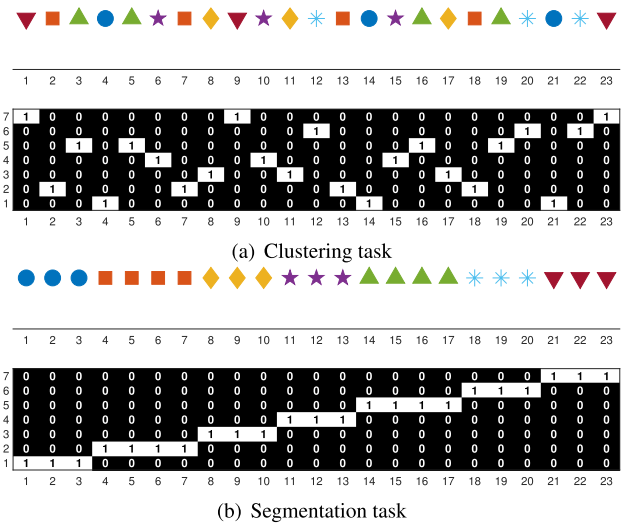
$$J_{KKM}(\boldsymbol{G}) = \operatorname{Tr}(\boldsymbol{L}\boldsymbol{K}), \tag{3}$$

where $\boldsymbol{L} = \boldsymbol{I}_n - \boldsymbol{G}^\top\left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1}\boldsymbol{G}$.

Kernel k-means is a strong approach for identifying clusters that are non-linearly separable in the original space. However, similar to its linear counterpart, kernel k-means is sensitive to outliers. More specifically, it often outputs unbalanced results that consists of too big and/or too small clusters under presents of anomaly data samples [40]. To alleviate this issue, recently [41] has proposed a simple regularization on the indicator matrix of the form $\operatorname{Tr}(\boldsymbol{G}\mathbf{1}\mathbf{1}^\top\boldsymbol{G}^\top)$, where $\mathbf{1}$ is a vector, whose all elements equal to one. By minimizing this regularization jointly with the clustering objective, we can prevent a too small or too great number of data samples from being partitioned into a cluster. We now can combine (3) and the regularization to form a new objective of balanced kernel k-means

$$J_{BKKM}(\boldsymbol{G}) = \operatorname{Tr}(\boldsymbol{L}\boldsymbol{K}) + \lambda\operatorname{Tr}(\boldsymbol{G}\mathbf{1}\mathbf{1}^\top\boldsymbol{G}^\top), \tag{4}$$

where λ is a positive parameter that controls the balanced regularization.



(a) Clustering task

(b) Segmentation task

FIGURE 2. Toy examples of (a) Clustering task and (b) Segmentation task, where the given data and the corresponding indicator matrix are depicted. Data samples from the same cluster or segment have identical symbol and color. Segmentation is different from clustering in that data samples of the same segment must be in a successive order.

IV. THE PROPOSED METHOD

A. KERNEL CLUSTERING WITH SIGMOID-BASED REGULARIZATION (KCSR)

Our intuitive idea is to reuse the robust objective of balanced kernel $k-$means (4) for segmentation of data sequence $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$. However, the challenge is that the sample-to-segment indicator matrix must satisfy two constraints, including *boundary* and *monotonicity*, while the indicator matrix for clustering does not. This difference is illustrated Figure 2. To close this gap and enable the clustering approach to segment data sequences, we introduce a novel regularization that smoothly approximates the two above constraints. The new regularization changes the variables from a discrete to continuous domains. Therefore, our problem can be solved using gradient descent (GD) algorithm. Since, the convergence of GD was already proved [22], quality of the proposed models' solutions is guaranteed.

The proposed regularization is based on the sigmoid function. A basic sigmoid function is defined as

$$f_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-\alpha(x-\beta)}}, \tag{5}$$

where β specifies the midpoint and α controls the steepness of the function curve at the midpoint. Figure 3 depicts a sigmoid function, where the midpoint β is fixed at 11.5 and the parameter α varies from 0.1 to 10.

We can observer that the higher α is the steeper function curve at the midpoint becomes. In addition, the sigmoid function is monotonic and almost piecewise constant. Therefore, it allows us to roughly partition a sequence into two segments, where the parameter β approximates the segment boundary. If we denote $\tau_j \in [1, 2]$ (continuously valued) as segment

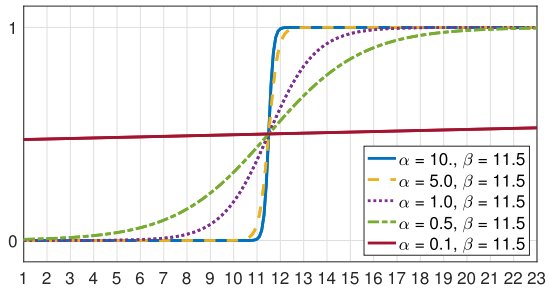


FIGURE 3. Sigmoid function with different values of the parameter α.

label of sample $\boldsymbol{x}_j$, then

$$\tau_j \approx 1 + f_{\text{sigmoid}}(j, \alpha, \beta). \tag{6}$$

For instance, if $\alpha = 10$ and $\beta = 11.5$, then $\tau_j \approx 1$ for $j < 11.5$ and 2 otherwise. To generalize for cases, where the number of segments $k > 2$, we propose to use a summation of $k-1$ sigmoid functions with different parameters β_i for $1 \le i \le k-1$.

$$\tau_j \approx 1 + \sum_{i=1}^{k-1} f_{\text{sigmoid}}(j, \alpha, \beta_i). \tag{7}$$

Figure 4 illustrates an example of a summation of sigmoid functions defined in (7). Here, the steepness parameter α is shared among the sigmoid functions within the summation. $k-1$ midpoint parameters $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{k-1}]$ approximate the segment boundaries between the k segments. Note that the midpoints must satisfy $1 \le \beta_1 < \ldots < \beta_{k-1} \le n$ to guarantee the summation of sigmoid functions monotonically increasing. Thus, we regularize the $\boldsymbol{\beta}$ by further introducing k parameters $\gamma_1, \ldots, \gamma_k$ such that

$$\beta_i = \left(1 - \frac{\sum_{i'=1}^{i} e^{\gamma_{i'}}}{\sum_{i'=1}^{k} e^{\gamma_{i'}}}\right) + n \times \frac{\sum_{i'=1}^{i} e^{\gamma_{i'}}}{\sum_{i'=1}^{k} e^{\gamma_{i'}}}. \tag{8}$$

In equation (8), the ratio $\frac{\sum_{i'=1}^{i} e^{\gamma_{i'}}}{\sum_{i'=1}^{k} e^{\gamma_{i'}}}$ is in the range $[0, 1]$. Therefore, β_i always satisfies $1 \le \beta_i \le n$. In addition, the ratio becomes larger as i increases. This guarantees that $\beta_{i'} < \beta_i$ for $1 \le i' < i \le k-1$.

It is notable that the summation of sigmoid functions in Figure 4 smoothly approximates the indicator matrix $\boldsymbol{G}$ of segmentation example in Figure 2(b). To make the observation more clear, we introduce the following approximation to each element of $\boldsymbol{G}$

$$G_{i,j} \approx \max\left(0, 1 - |\tau_j - i|\right). \tag{9}$$

This equation map the segment label τ_j from the range $[1, k]$ to the range $[0, 1]$ for approximating the sample-to-segment indicator matrix.

We now can formulate an optimization problem that combines objective of the balanced kernel clustering with sigmoid-based regularization for segmentation.

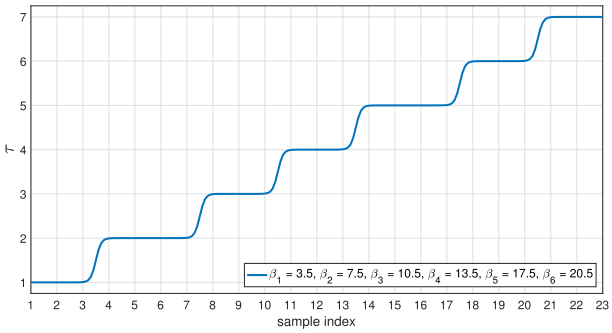


FIGURE 4. An example of the summation of sigmoid functions with a shared parameter $\alpha = 10$ and $k-1$ different midpoint parameters $\beta_1, \ldots, \beta_{k-1}$, where $k = 7$.

Algorithm 1 Gradient Descent Algorithm for KCSR

Require: Kernel matrix $\boldsymbol{K}$, number of segments k, steepness parameter α, tolerance ϵ.
Ensure: Optimal parameters $\boldsymbol{\gamma}^* = [\gamma_1^*, \ldots, \gamma_k^*]^\top$.
1: **repeat**
2: compute gradient $\nabla\boldsymbol{\gamma} = \frac{\partial J}{\partial \boldsymbol{\gamma}}$;
3: compute stepsize η using Armijo-Goldstein line search [22], [42];
4: update $\boldsymbol{\gamma}_{(t+1)} = \boldsymbol{\gamma}_{(t)} - \eta\nabla\boldsymbol{\gamma}_{(t)}$;
5: **until** $|J(\boldsymbol{\gamma}_{(t+1)}) - J(\boldsymbol{\gamma}_{(t)})| \le \epsilon$

Let $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ be the kernel matrix of the data sequence $\boldsymbol{X}$ then our kernel-based segmentation optimization problem is

$$\begin{aligned}
&\underset{\gamma_1,\ldots,\gamma_k}{\text{argmin}} \ \ \text{Tr}\,(\boldsymbol{LK}) + \lambda\,\text{Tr}(\boldsymbol{G}\mathbf{1}\mathbf{1}^\top\boldsymbol{G}^\top) \\
&\text{s.t. } \boldsymbol{L} = \boldsymbol{I}_n - \boldsymbol{G}^\top\left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1}\boldsymbol{G}, \\
&\quad G_{i,j} = \max\left(0, 1 - |\tau_j - i|\right) \quad \forall i, j, \\
&\quad \tau_j = 1 + \sum_{i=1}^{k-1} f_{\text{sigmoid}}(j, \alpha, \beta_i) \quad \forall j, \\
&\quad \beta_i = \left(1 - \frac{\sum_{i'=1}^{i} e^{\gamma_{i'}}}{\sum_{i'=1}^{k} e^{\gamma_{i'}}}\right) + n \times \frac{\sum_{i'=1}^{i} e^{\gamma_{i'}}}{\sum_{i'=1}^{k} e^{\gamma_{i'}}} \quad \forall i.
\end{aligned} \tag{10}$$

Since $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_k]$ are unconstrained and continuous parameters, we can optimize objective function in (10) using the gradient descent algorithm. Let $J(\boldsymbol{\gamma})$ denotes the objective function in (10), then the gradient w.r.t parameters $\boldsymbol{\gamma}$ can be computed using chain rule.

$$\nabla\boldsymbol{\gamma} = \frac{\partial J(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}} = \frac{\partial J(\boldsymbol{\gamma})}{\partial \boldsymbol{G}} \times \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{\tau}} \times \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\beta}} \times \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\gamma}}, \tag{11}$$

where $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_n]$. More details on derivation of the gradient w.r.t $\boldsymbol{\gamma}$ is given in Appendix. We call the proposed model *Kernel clustering with sigmoid regularization* (KCSR) and its optimization algorithm is given in Algorithm 1.

B. STOCHASTIC KCSR

Kernel segmentation allow us to capture nonlinear structure in the data. However, this advantage is achieved at the expense

TABLE 1. Time and space complexities of different segmentation methods. Here, n denotes length of the data sequence and k is the number of segments. $n_{\max}$ denotes the maximum length of divided subsequences in ACA, d is dimension of the new representation Z in OSC and TSC. t denotes number of total iterations. The rank of the approximation of the kernel matrix in AKS is denoted by r and b is the mini-batch size in SKCSR. Note that $b \ll n$.

Method	SSC	TSC	ACA	AKS	GKS	**KCSR**	**SKCSR**
Time	$O(n^2dt + n^2)$	$O(n^2dt + n^2)$	$0(n^2n_{\max}t)$	$O(r^2n + r\log(k)n)$	$O(kn + n^2)$	$O(n^2kt + n^2)$	$O(nbkt + b^2)$
Space	$O(n^2)$	$O(n^2)$	$O(n^2n_{\max})$	$O((k + r)n)$	$O(n^2)$	$O(n^2)$	$O(b^2)$

of much higher complexities in both terms of computational time and storage requirement. More specifically, given a sequence of n data samples, existing kernel-based methods compute the kernel matrix $\boldsymbol{K}$, whose both time and memory complexities are of order $O(n^2)$. Note that this is also true for temporal clustering methods, where the affinity graph matrix of size $O(n^2)$ is computed and stored while performing Ncut algorithm. When n is large, these methods become computationally difficult. For example, average length of the acceleration data for activity recognition in the experimental section is about $125K$. The corresponding kernel matrix $\boldsymbol{K}$ requires up to approximately 116.4 GB for storage, which is definitely out of memory for a regular computer.

Our method is also based on the kernel matrix. Especially, at each iteration, our method computes the gradient using the kernel matrix, which makes it very slow and even impossible due to the large memory requirement for handling long data sequences. Fortunately, since objective function of KCSR is differentiable, we can reduce the complexities by using the stochastic gradient descent (SGD) [43]–[45]. SGD estimates the gradient from a randomly sampled subsequence[2] (a mini-batch), which consists of a much smaller number of samples, from the original sequence. Let $b \ll n$ denotes length of the randomly sampled subsequence $\boldsymbol{X}_{(t)}$, where t expresses the iteration index. Then, the stochastic gradient is estimated as follows

$$\nabla\boldsymbol{\gamma} = \frac{\partial J(\boldsymbol{\gamma})}{\partial \boldsymbol{G}_{(t)}} \times \frac{\partial \boldsymbol{G}_{(t)}}{\partial \boldsymbol{\tau}} \times \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\beta}} \times \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\gamma}}. \tag{12}$$

In equation (12), $\frac{\partial J(\boldsymbol{\gamma})}{\partial \boldsymbol{G}_{(t)}}$ is only associated with a partial kernel matrix $\boldsymbol{K}_{(t)} \in \mathbb{R}^{b\times b}$, which corresponds to the samples in $\boldsymbol{X}_{(t)}$. Therefore, it is much more efficient in terms of both running time and memory consumption than computing the full-batch gradient as in equation (11). Details of the algorithm is given in Algorithm 2 and complexity comparison between the proposed methods and several baselines are given in Table 1. We note that convergences of both gradient with step size found by Armijo-Goldstein line search [22], [42] and stochastic gradient descent algorithms with vanishing step size are theoretically proven. In fact, it is well-known [46], [47] that gradient descent (GD) after T iterations can find a solution with error $O(T^{-1})$ and stochastic gradient descent (SGD) after T iterations can find a solution with error $O(T^{-0.5})$. Thus, both KCSR and SKCSR can obtain good solutions for problem defined in (10) with enough loops.

[2]By sub-sequence, we mean that order and indexes of samples in the original sequence are preserved in the randomly sampled mini-batch.

Algorithm 2 Stochastic Gradient Descent Algorithm for KCSR

Require: Data sequence $\boldsymbol{X}$, number of segments k, steepness parameter α, number of iterations T, minibatch size b, initial learning rate η_0, momentum $\mu \in [0, 1)$, weight decay $\rho \in (0, 1]$.

Ensure: Optimal parameters $\boldsymbol{\gamma}^* = [\gamma_1^*, \ldots, \gamma_k^*]^\top$.

1: **for** $t = 1, \ldots, T$ **do**
2: $\quad\eta = \eta_0 \times \rho^t$;
3: $\quad$randomly sample a sub-sequence $\boldsymbol{X}_{(t)}$ of length b;
4: $\quad$compute the partial kernel matrix $\boldsymbol{K}_{(t)}$;
5: $\quad$compute the stochastic gradient $\nabla\boldsymbol{\gamma} = \frac{\partial J}{\partial \boldsymbol{\gamma}}$ based on $\boldsymbol{K}_{(t)}$ and original indexes of samples in $\boldsymbol{X}_{(t)}$;
6: $\quad\Delta\gamma_{(t)} = \eta\nabla\boldsymbol{\gamma} - \mu\Delta\gamma_{(t-1)}$;
7: $\quad\boldsymbol{\gamma}_{(t)} = \boldsymbol{\gamma}_{(t-1)} + \Delta\gamma_{(t)}$;
8: **end for**

C. MULTIPLE KCSR

In practice, at some particular circumstances, we need to perform segmentation on multiple data sequences. If these sequences are not in relation, the problem is effortless since segmentation algorithms can be applied on each sequence independently. However, when the sequences are related to each other, performing multiple segmentation without considering relation among the sequences would induces inferior results. We take sequential segmentation and matching (SSM) problem as a study case. Given $m \geq 2$ data sequences, SSM aims at partitioning each sequence into several homogeneous segments and then establishing the correspondences between these segments from different sequences. A popular application of SSM is human action analysis. Specifically, the human action videos are segmented into primitive actions and the resulted sequences of the action segments are then aligned [48]–[50].

To solve the SSM problem, in this work, we introduce an extension of the proposed model termed *Multiple kernel clustering with sigmoid-based regularization* (MKSSR). MKSSR jointly partitions each data sequences into k segments such that the c^{th} segments of all the m sequences are matched.[3] Let $\boldsymbol{X}_p \in \mathbb{R}^{d\times n_p}$ for $1 \leq p \leq m$ denotes the p^{th} data sequence and $\boldsymbol{G}_p \in \mathbb{R}^{k\times n_p}$ be its corresponding sample-to-segment indicator matrix. MKSSR firstly concatenates all the sequences to form a single long sequence $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m] \in \mathbb{R}^{d\times n}$, where $n = \sum_{i=1}^{m} n_p$. Then $\boldsymbol{G} = [\boldsymbol{G}_1, \ldots, \boldsymbol{G}_m] \in \mathbb{R}^{k\times n}$ is the

[3]Data samples of the c^{th} segments from different sequences belong to the c^{th} class for $1 \leq c \leq k$.

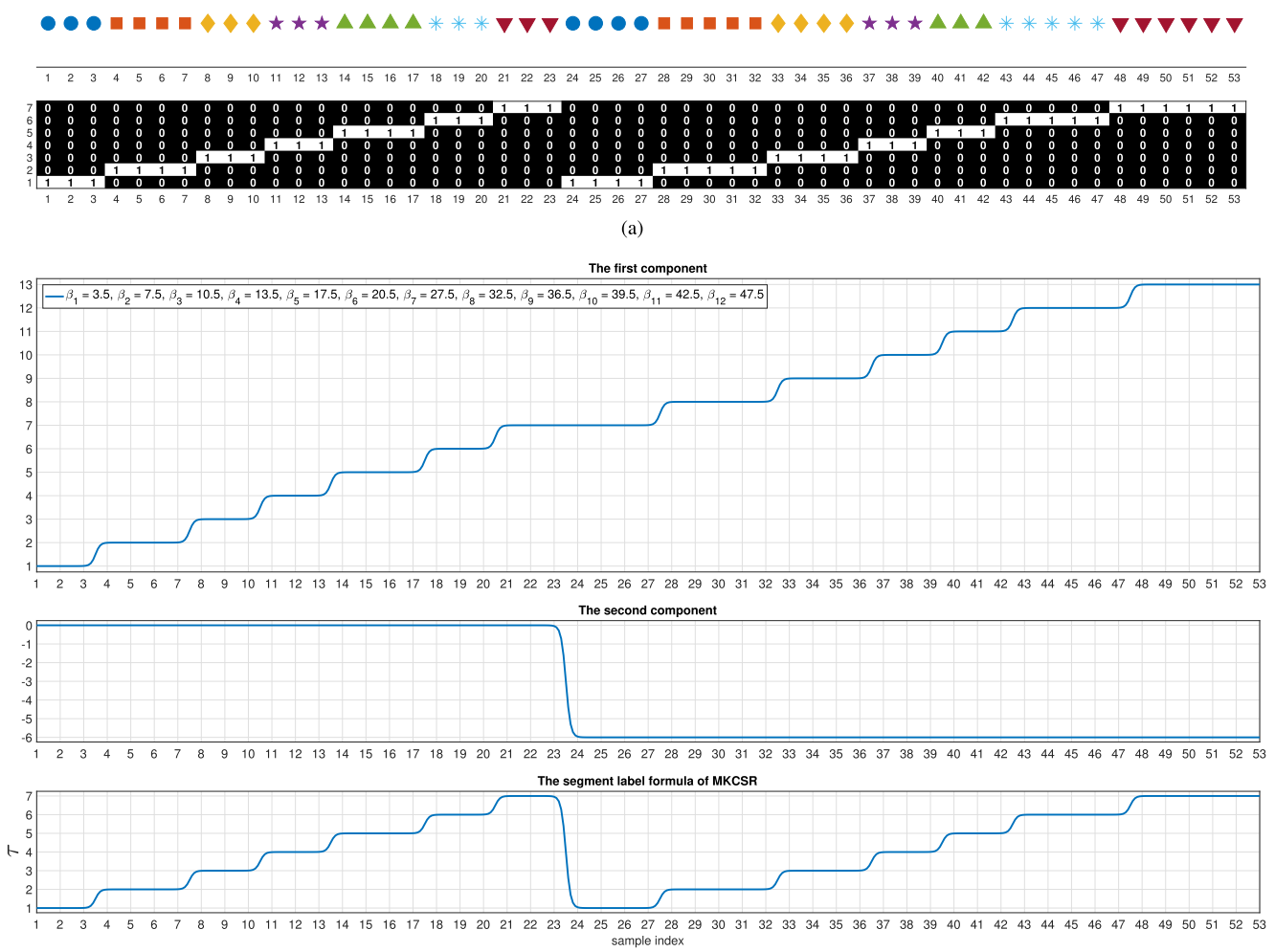


FIGURE 5. Illustration of the cut-off summation of sigmoid functions. (a) A toy example of a concatenation of two sequences ($m = 2, n_1 = 23, n_2 = 30$) and its corresponding indicator matrix ($k = 7$). (b) The cut-off summation of sigmoid functions, whose two components are depicts in the two first subfigures, can smoothly approximate the indicator matrix in the toy example.

corresponding indicator matrix of $\boldsymbol{X}$. Similar to the original KCSR, each element of $\boldsymbol{G}$ is defined as in (9). However, in MKCSR, the segment label τ_j is computed as following

$$\tau_j = 1 + \sum_{i=1}^{m(k-1)} f_{\text{sigmoid}}(j, \alpha, \beta_i) + (1-k)\sum_{p=1}^{m-1} f_{\text{sigmoid}}(j, \alpha, \sum_{q=1}^{p} n_q + 0.5). \quad (13)$$

The function (13), which we call as cut-off summation of sigmoid functions, consists of two components. The first component is the summation of sigmoid functions. It plays a similar role as (7) in KCSR. The second component presents the cut-off points (a.k.a junction points), at which two among the m original data sequences are connected. It will reset the segment label from k to 1 after passing the final sample of one sequence and reaching a new sample from the next sequence. The cut-off summation of sigmoid functions and its components are illustrated in Figure 5.

The formulation (13) has $m(k-1)$ midpoint parameters, in which $\beta_{(p-1)(k-1)+1}, \ldots, \beta_{p(k-1)}$ approximate the segment boundaries within the range $[1 + \sum_{q=1}^{p-1} n_q, \sum_{q=1}^{p} n_q]$ for $1 \leq p \leq m$. Therefore, we introduce mk parameters $\gamma_1, \ldots, \gamma_{mk}$ such that for $(p-1)(k-1)+1 \leq j \leq p(k-1)$

$$\beta_i = \left(1 + \sum_{q=1}^{p-1} n_q\right)\left(1 - \frac{\sum_{i'=(p-1)k+1}^{i} e^{\gamma_{i'}}}{\sum_{i'=(p-1)k+1}^{pk} e^{\gamma_{i'}}}\right) + \sum_{q=1}^{p} n_q \frac{\sum_{i'=(p-1)k+1}^{i} e^{\gamma_{i'}}}{\sum_{i'=(p-1)k+1}^{pk} e^{\gamma_{i'}}}. \quad (14)$$

By replacing the last two constraints in (10) with (13) and (14) we can obtain the optimization problem of MKCSR. The objective function is then minimized w.r.t mk parameters $\gamma_1, \ldots, \gamma_{mk}$ using the stochastic gradient descent algorithm.

V. EXPERIMENTS

A. BASELINES

We compare KCSR and its stochastic variant SKCSR with the following baselines

- Aligned clustering analysis (ACA) [30] – a temporal clustering method that combines $k-$means with Dynamic time alignment kernel [51].
- Sequential subspace clustering (SSC) [1] – a temporal clustering method that combines subspace clustering with linearly temporal regularization weighted by ℓ_1-norm sequential graph.
- Temporal subspace clustering (TSC) [2] – a temporal clustering method that combines subspace clustering with manifold-based temporal regularization and low-rank constraint.
- Approximate kernel segmentation (AKS) [20] – a heuristic approximation of the optimal kernel segmentation, where the solution is obtained by pruned DP algorithm that combines a low-rank approximation of the kernel matrix and the binary segmentation algorithm.
- Greedy kernel segmentation (GKS) [21] – another heuristic approximation of the optimal kernel segmentation that detects the segment boundaries sequentially using greedy algorithm.

B. DATASETS

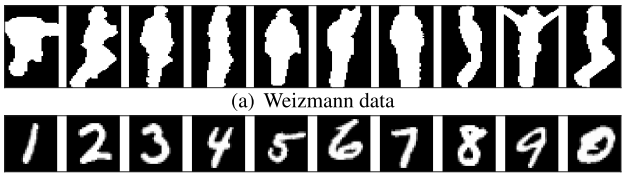
To evaluate performances of the above methods, we use a synthetic dataset and five real-world and widely public datasets.

1) SYNTHETIC DATA

We first generate $2D$ data samples that form four circles of different diameters. They are illustrated in Figure 8(a). The number of data samples of each circle is randomly selected in range [500, 1500] and also constrained to be different. For instance, in our case, the numbers of data samples of the circles from low to high diameters are 832, 1018, 1174 and 843, respectively. We then rearrange the generated data samples in contiguous order, i.e. data samples of one circle do not mix to the other circles. By doing so, each circle in the original $2D$ space corresponds to a segment in the new sequential data. See Figure 8(b) for illustration.

2) WEIZMANN DATA

The Weizmann dataset [52] consists of 90 videos of nine subjects, each performing ten actions: bend, run, jump-in-place (pjump), walk, jack, wave-one-hand (wave1), side, jump-forward (jump), wave-two-hand (wave2), and skip. Similar to [53], videos of the same subjects are concatenated into a long video sequence following the presented order of the actions. We then subtract background from each frame of these video sequences and rescale them to the size 70×35. For each $70-$by-35 rescaled frame, we compute the binary feature as shown in Figure 6(a). To reduce the dimensions of the feature space (2450), the top 123 principal components that preserve 99% of the total energy are kept for experiments.



(a) Weizmann data

(b) Ordered MNIST data

FIGURE 6. (a) Concatenated action videos of subject 1 in Weizmann dataset and (b) the rearranged digit images sequence in MNIST dataset. Each data sequence consists of 10 non-overlapping segments and only one representative frame of each segment is depicted.

3) MMI FACIAL ACTION UNITS

We exploit the MMI Facial Expression dataset [54], which contains more than 2900 videos of 75 different subjects, each performing a particular combination of Action Unit (AU). In this paper, we focus on videos of AU12, which corresponds to a smile. Although, these videos consist of different number of frames, they are composed of exactly five segments with the following order: neutral, onset, apex, offset, neutral, where *neutral* is when facial muscle is inactive, *apex* is when facial muscle intensity is strongest, and *onset* is when facial muscle starts to activate or *offset* is when facial muscle begins to relax. Following the same pre-processing procedure as in [55], we cropped and aligned the face using dlib-ml [56]. The results are depicted in Figure 7. We then convert them to grayscale and reduce their dimension to 400 using whitening PCA. We finally selected videos of five subjects 2, 3, 6, 14 and 17 for experiments. Their ground-truth frames-to-segment lables are already given in the original dataset.

4) GOOGLE SPOKEN DIGITS

Google's Speech Commands (GSC) [57], [58] is a large audio dataset that consists of more than 30 categories of spoken terms. For each category that relates to digits from ''one'' to ''nine'', we randomly select a clean recording. These recordings are then concatenated, forming a long audio sequence with 19 segments (9 segments of active voice and 10 silent segments) (see Fig. 10). We further add white noise, which is also provided in the GSC dataset, to make the segmentation problem more challenging. Finally, a sequence of acoustic features, which are 13-dimensional mel–frequency cepstral coefficients (MFCCs) [59] for every 10ms of a 25ms window, is computed from the noisy audio sequence. The annotation is manually obtained based on the log filter-bank energies of the clean audio.

5) ORDERED MNIST DATA

the MNIST dataset [60] consists of 28×28 grayscale digit [0, 9] images divided into $60K/10K$ for training/testing. Since all the compared methods are unsupervised and require no training phase, we use all $70K$ images to perform segmentation. Note that the original data is not exact suited to the sequential assumption. Following the same setting of [2], we rearrange order of the images such that those of the same digit form a contiguous segment and the ten segments

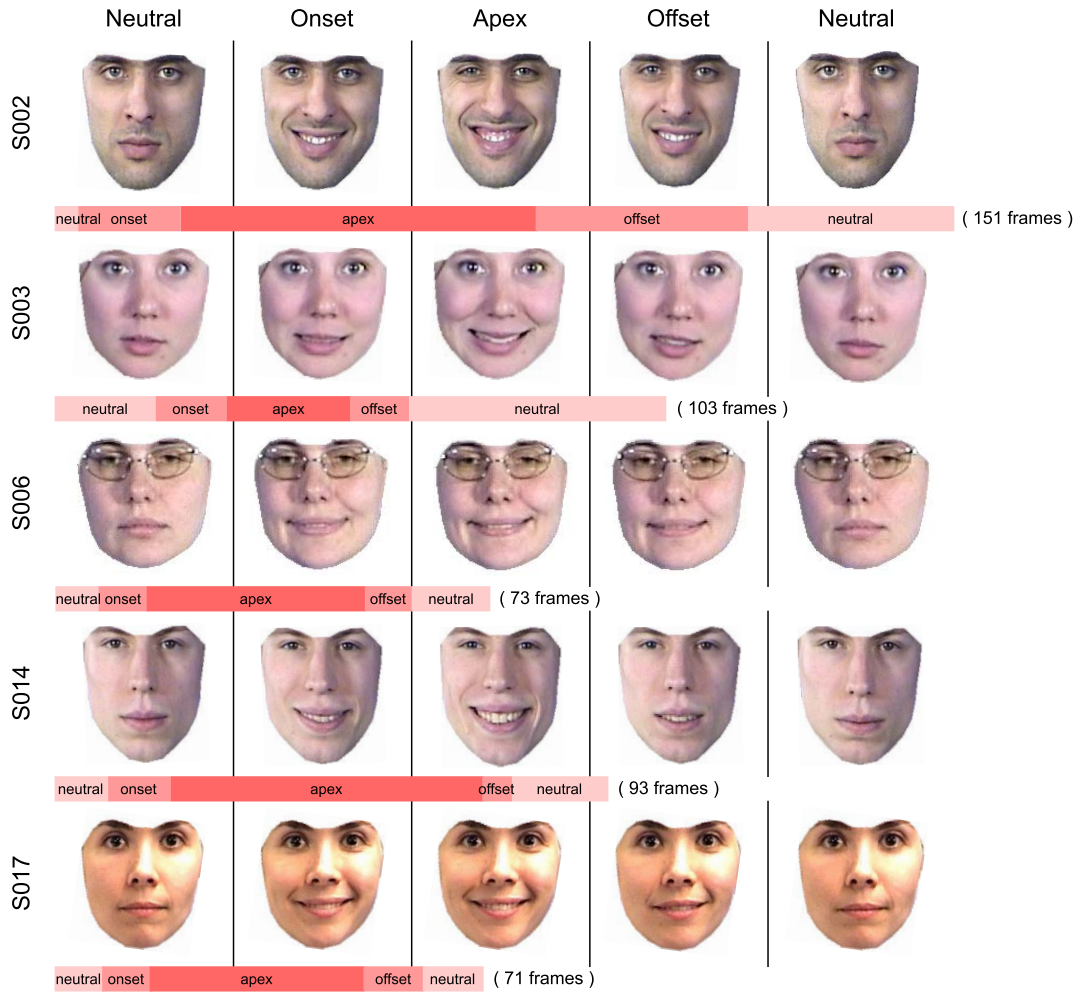


FIGURE 7. Videos of five subjects (S002, S003, S006, S014 and S017) performing an action unit 12 that corresponds to smile taken from MMI Facial action units dataset. The representative facial images of the segments are depicted. The bottom of each video shows duration of the corresponding ground truth frame-to-segment labels along with the total number of frames.

are concatenated into a very long images sequence (see Figure 6(b)). Different from [2], where only $2K$ images were selected for experiment, our ordered MNIST data consists of the whole $70K$ images. To handle this large-scale data, temporal clustering and kernel CPD methods requires up to 36.5 GB to store the kernel and/or affinity graph matrices, which is impractical for implementation on a single personal PC. Among the compared methods, only SKCSR and AKS with low memory complexities can perform segmentation on this dataset.

6) ACCELERATION DATA[4]

The acceleration data [61] are acquired from a triaxial accelerometer mounted on the chests of 15 subjects, each performing a sequence of activities such as working at computer, standing, walking, going updown stairs and talking. The aims of our experiments is to partition the data sequences into segments that correspond to the activities. Thus, we firstly pre-process the data. For each subject, we add squares of signals from the three axles. An example is depicted in Figure 11. We then transform the obtained summation signal using wavelet transform with scale factor 64 and the Morlet wavelet as the mother wavelet function. The resulting 2D wavelet coefficient matrix $\boldsymbol{C}$ is of the size 64-by-n_{acc}, where n_{acc} is the length of the original acceleration signal. Note that the wavelet coefficients C are complex numbers. Thus we take its modulus as input for the methods in our experiments. Similar to the Ordered MNIST data, this dataset consist of long acceleration sequences. The average n_{acc} is $125K$. Therefore, the methods with memory complexities of order $O(n^2)$ will require up to approximately 116.4 GB, which is unaffordable in our case, for storage. In our experiments, only SKCSR and AKS can handle this dataset.

[4]The acceleration dataset is available in UCI repositories and can be retrieved from https://archive.ics.uci.edu/ml/machine-learning-databases/00287/.

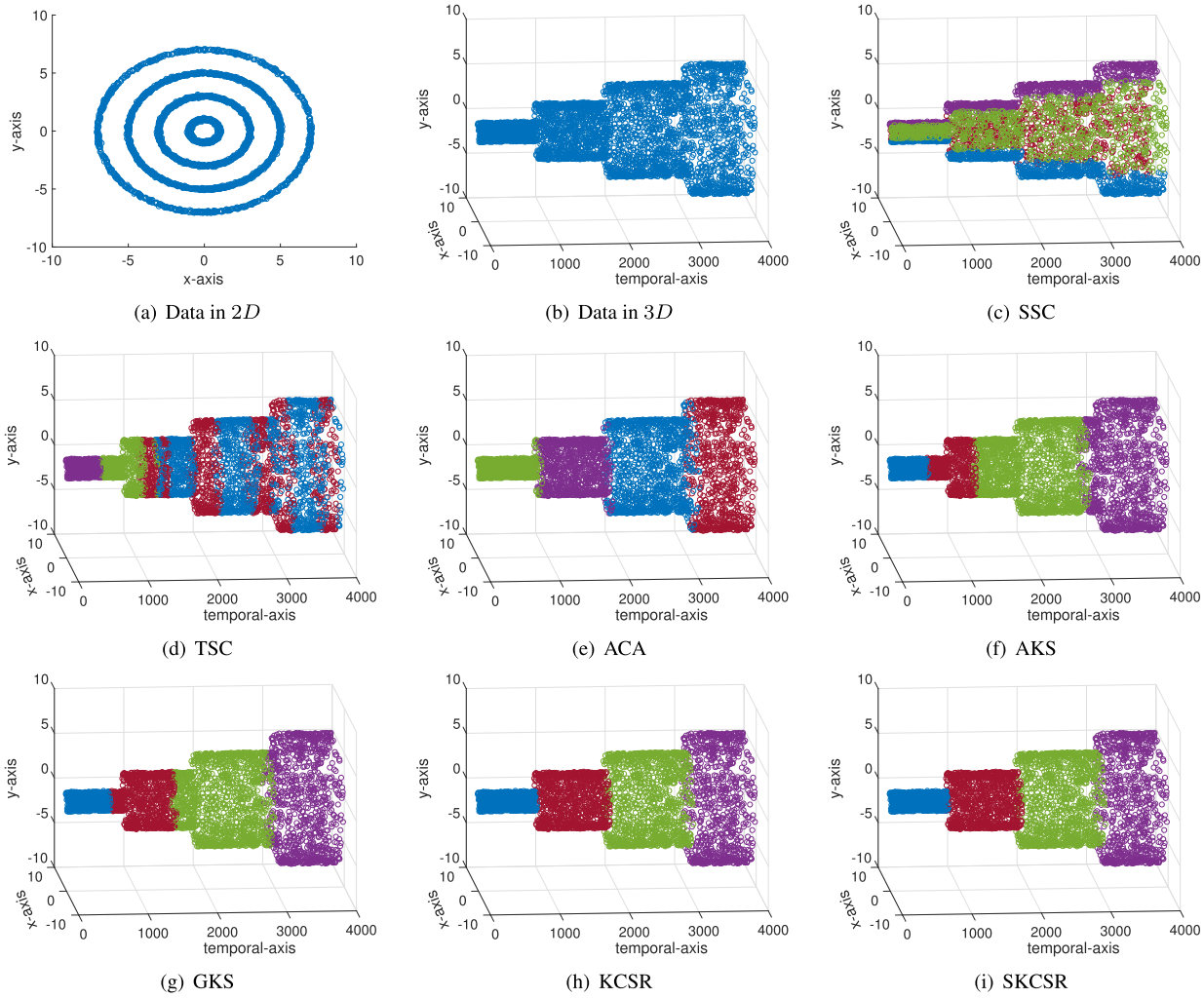


FIGURE 8. Synthetic experiment: (a) data generated in 2*D* space, (b) the data after contiguously rearranging and visualization of segmentation results returned by all the compared methods. Different colors represent different clusters.

C. EVALUATION MEASURES

Given a specific value k, while KCSR, SKCSR, AKS and GKS return exactly k non-overlapping segments, temporal clustering-based methods partition samples of the data sequence into k clusters that maybe dispersed in discontiguous segments. Since, all the compared methods base on clustering scheme, we use accuracy and normalized mutual information [62] as evaluation metrics to assess the segmentation results.

Let $\widehat{\boldsymbol{\mathcal{L}}} = [\hat{l}_1, \ldots, \hat{l}_n]$ and $\boldsymbol{\mathcal{L}} = [l_1, \ldots, l_n]$ be the obtained labels and ground-truth labels of a given data sequence $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$. $\hat{l}_j = i$ (similar for l_j) for $1 \leq i \leq k$ indicates that $\boldsymbol{x}_j$ belongs to cluster (segment) $\hat{c}_i$. The accuracy (ACC) is defined as follows:

$$ACC = \frac{\sum_{j=1}^{n} \delta(l_j, map(\hat{l}_j))}{n}, \tag{15}$$

where $\delta(a, b)$ is the delta function that equals one if $a = b$ and zero otherwise and $map(\hat{l}_j)$ is the permutation mapping function that maps label $\hat{l}_j$ to the equivalent ground truth label. In this work, we use Kuhn-Munkres algorithm [63] to find the mapping.

Let $\widehat{\mathcal{C}} = [\hat{c}_1, \ldots, \hat{c}_k]$ and $\mathcal{C} = [c_1, \ldots, c_k]$ be the obtained clusters and the ground-truth clusters. Their mutual information (MI) is

$$MI(\mathcal{C}, \widehat{\mathcal{C}}) = \sum_{c_i \in \mathcal{C}, \hat{c}_{i'} \in \widehat{\mathcal{C}}} p(c_i, \hat{c}_{i'}) \log_2 \frac{p(c_i, \hat{c}_{i'})}{p(c_i)p(\hat{c}_{i'})}, \tag{16}$$

where $p(c_i)$ and $p(\hat{c}_{i'})$ are the probabilities that a data sample arbitrarily selected from the sequence belongs to the clusters c_i and $\hat{c}_{i'}$, respectively, and $p(c_i, \hat{c}_{i'})$ is the joint probability that the selected data sample belongs to both c_i and $\hat{c}_{i'}$. This metric is normalized to the range [0, 1] as follows:

$$NMI(\mathcal{C}, \widehat{\mathcal{C}}) = \frac{MI(\mathcal{C}, \widehat{\mathcal{C}})}{max(H(\mathcal{C}), H(\widehat{\mathcal{C}}))}, \tag{17}$$

where $H(\mathcal{C})$ and $H(\widehat{\mathcal{C}})$ are the entropies of $\mathcal{C}$ and $\widehat{\mathcal{C}}$, respectively.

TABLE 2. Segmentation results on six datasets, including synthetic data, Weizmann action sequences, MMI Facial smiling video, noisy Google spoken digits, ordered MNIST data and Acceleration sequences, returned by different methods. The mean score of each methods over five random runs along with its variance are reported. The symbol "–" means that there is no result due to the shortage of memory resources.

Dataset		SSC	TSC	ACA	AKS	GKS	KCSR	SKCSR
Synthetic data	ACC	0.0965 (0.0736)	0.4077 (0.0795)	0.9305 (0.0224)	0.6577 (0.0671)	0.6999 (0.0255)	**0.9871** (0.0104)	0.9870 (0.0092)
	NMI	0.1070 (0.0649)	0.3608 (0.0758)	0.9214 (0.0375)	0.6067 (0.0751)	0.7036 (0.0325)	**0.9959** (0.0024)	0.9847 (0.0077)
Weizmann	ACC	0.4856 (0.0269)	0.7028 (0.0430)	0.7687 (0.0146)	0.7182 (0.0188)	0.5628 (0.0218)	0.8835 (0.0092)	**0.8964** (0.0113)
	NMI	0.4157 (0.0387)	0.7207 (0.0501)	0.7628 (0.0196)	0.7006 (0.0205)	0.6032 (0.0262)	0.9071 (0.0107)	**0.9151** (0.0095)
MMI Facial AU	ACC	0.6453 (0.0238)	0.7552 (0.0392)	0.8121 (0.0157)	0.7527 (0.0204)	0.6025 (0.0249)	0.9537 (0.0118)	**0.9763** (0.0075)
	NMI	0.6514 (0.0412)	0.7321 (0.0437)	0.7952 (0.0205)	0.7600 (0.0214)	0.6355 (0.0298)	0.9625 (0.0129)	**0.9686** (0.0126)
Google	ACC	0.2057 (0.0275)	0.6434 (0.0313)	0.8241 (0.0182)	0.6726 (0.0257)	0.7458 (0.0294)	0.7914 (0.0172)	**0.8826** (0.0165)
	NMI	0.1839 (0.0256)	0.6513 (0.0330)	0.7939 (0.0196)	0.6954 (0.0266)	0.7557 (0.0305)	0.8109 (0.0154)	**0.9009** (0.0186)
Ordered MNIST	ACC	–	–	–	0.6983 (0.0282)	–	–	**0.9681** (0.0155)
	NMI	–	–	–	0.7196 (0.0209)	–	–	**0.9819** (0.0119)
Acceleration	ACC	–	–	–	0.5535 (0.0434)	–	–	**0.8172** (0.0216)
	NMI	–	–	–	0.5763 (0.0398)	–	–	**0.8056** (0.0311)

D. PARAMETER SETTINGS

We select the optimal parameters for each method to achieve the best performance. The number of clusters k of all the compared methods is set to the number of segments available in the datasets. For ACA, its parameters *nMa* and *nMi* that specify the maximum and minimum lengths of each divided subsequence, respectively, are data-dependent. Let n be the sequence length, we select *nMa* from a rounded set $\{0.01n, 0.02n, 0.04n, 0.06n, 0.08n, 0.1n\}$ and set $nMi = \frac{nMa}{2}$. For temporal subspace clustering methods, including SSC and TSC, the most important parameter is that controls the sequential regularization for the new representation $\boldsymbol{Z}$. We select this parameter from the set $\{1, 5, 10, 15, 20, 25\}$ and the other parameters are set according to the original papers. For the proposed methods, we fix the parameter that controls the steepness of the summation of sigmoid functions at the midpoints $\alpha = 10$. The tolerance ϵ for convergence verification in KCSR is fixed at 10^{-6}. For all the datasets, we use the Radial Basis Function (RBF) Kernel[5] with proper width σ for AKS, GKS and the proposed methods. The minibatch size b of SKCSR and the rank r of the approximation of the kernel matrix in AKS are kept equal. Their values are selected from a set $\{64, 128, 256, 512, 1024, 2048\}$. Note that, SKCSR terminates after processing T minibatches. We set T such that $T \times b \geq 50n$ (passing through the data sequence at least 50 times).

E. RESULTS DISCUSSION

1) EVALUATION OF KCSR AND SKCSR

Figure 8 visualizes the segmentation results on synthetic data and the evaluation scores are given in the first rows of Table 2. We can observe that each segment of the generated data sequence has a circular structure. Therefore, the nonlinear regularization in sequential representation learning of SSC is ineffective on the synthetic data. TSC performed significantly better. The manifold-based regularization allows it to be able to capture the nonlinear structure in the data. Our methods also perform segmentation based on regularization.

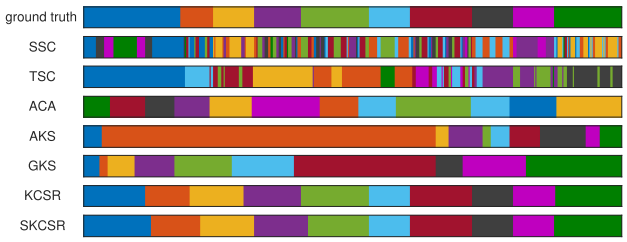


FIGURE 9. Visualization of segmentation results returned by the proposed methods and baselines on Weizmann dataset. Different colors represent different clusters.

However, different from SSC and TSC, where the regularization is just local,[6] the summation of sigmoid functions of KCSR and SKCSR globally regularizes the whole data sequences and the locality is ensured by its smooth nature. Therefore, the proposed methods obtained the best performance on the synthetic dataset.

On the real-world data, including Weizmann action videos, MMI Facial smiling videos and Google spoken digits audio, the proposed models also outperformed the baselines. Evaluation scores of the corresponding segmentation results are shown in the second, third and fourth rows of Table 2. We can observe that ACA also had good performances on these datasets. Although ACA also performs segmentation based on clustering as our methods do, it cannot guarantee to find exact k non-overlapping segments. Therefore, its evaluation scores are slightly lower than those of the proposed models. In comparison with heuristic approximations AKS and GKS, our models also had better performances. Similar to AKS and GKS, our models also search for segment boundaries. They approximate the boundaries by midpoints $\boldsymbol{\beta}$ of the summation of sigmoid functions. However, different from these heuristic approximations that search for the segment boundaries sequentially, the proposed models simultaneously obtain all the $\boldsymbol{\beta}$ via gradient-based algorithm. As convergence of this optimization algorithm is theoretically proved, optimality of the solutions is guaranteed. To qualitatively assess the

[5] RBF kernel: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2}{2\sigma^2}}$.

[6] The regularization only preserves the local relationship on representation of consecutive samples.

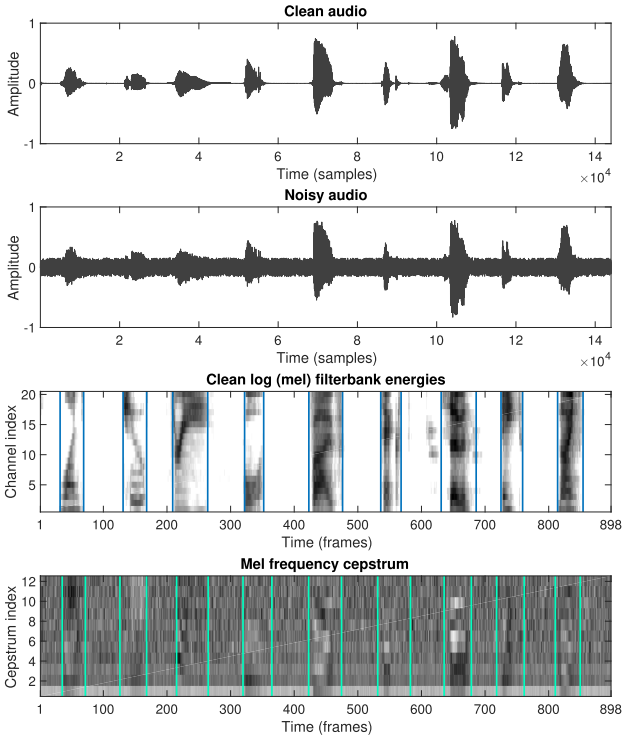


FIGURE 10. From the top to the bottom: clean audio of spoken digits [1, 9], the audio contaminated by white noise, log filter-bank energies of the clean audio used for manual annotation (blue lines depict ground truth segment boundaries) and Mel-frequency cepstrum of the noisy audio (vertical lines show the midpoints β of the summation of sigmoid functions returned by SKCSR).

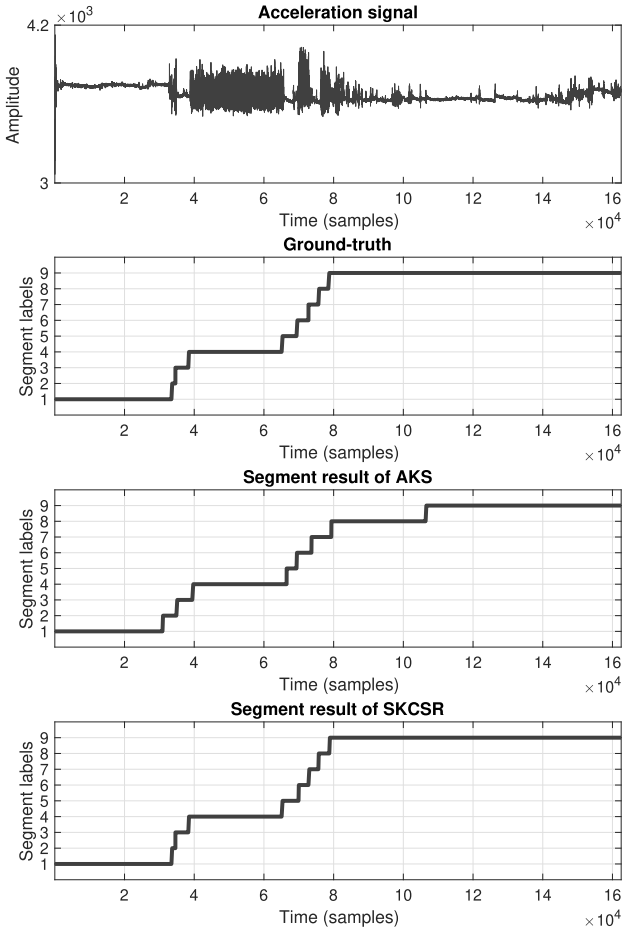


FIGURE 11. From the top to the bottom: Acceleration signal of the first subject, the corresponding ground-truth segment labels and the segmentation results returned by AKS and SKCSR, respectively, on the Acceleration dataset.

performances of the compared methods, we also visualized the segmentation results on Weizmann video and Google audio datasets in Figure 9 and Figure 10, respectively. These visualization further validate the superior performances of our methods over those of the baselines.

On these datasets, we also observe that evaluation scores of SKCSR are greater than those of KCSR. Thus, we further investigate convergence curves of these models. Figure 13 depicts those of SKCSR and KCSR on Weizmann action videos and Google spoken digits audios, respectively. It is clear that superior performances of SKCSR arise from the exploitation of stochastic gradient descent (SGD) algorithm. SGD allows SKCSR to update its parameter $\boldsymbol{\gamma}$ more frequently due to fast estimation of the stochastic gradient. In addition, SGD takes randomness of the data into account and enjoys theoretical guarantee on convergence in an expectation sense [44]. Therefore, SKCSR is more robust to noise in the data and able to achieve better solution than KCSR.

SKCSR also showed its superior efficiency over the original KCSR and most the other baselines on the ordered MNIST and Acceleration data. Recall that the ordered MNIST data consists of $70K$ samples. Acceleration data contains even much more longer data sequences, where the average length is $125K$. This makes implementation of the memory-demanding methods impossible on regular personal PCs. Among the baselines, only AKS with memory complexity of order $O(r^2)$, where $r \ll n$ is the rank of approximation of the kernel matrix, can handle the ordered MNIST and Acceleration data. However, since AKS employs binary segmentation to sequentially detect the segment boundaries, its solutions are not optimally guaranteed. Visualization of the segmentation results on the Acceleration data in Fig. 11 and the evaluation scores in the fifth and sixth rows of Table 2 validate the advantages of SKCSR.

2) EVALUATION OF MKCSR

We next evaluate performance of MKCSR – an extension of KCSR for handling multiple data sequences. We utilize concatenated Weizmann action videos and MMI Facial AU videos in this experiment. For the Weizmann data, the first, second and third subjects are selected and their corresponding action videos are concatenated to form a long sequence that consists of 30 segments, each of which belong to one of the ten action categories. For the MMI Facial AU data, videos of all the subjects are concatenated. The new video sequences consists of 491 frames and 15 segments. We compare MKCSR with temporal clustering methods, including

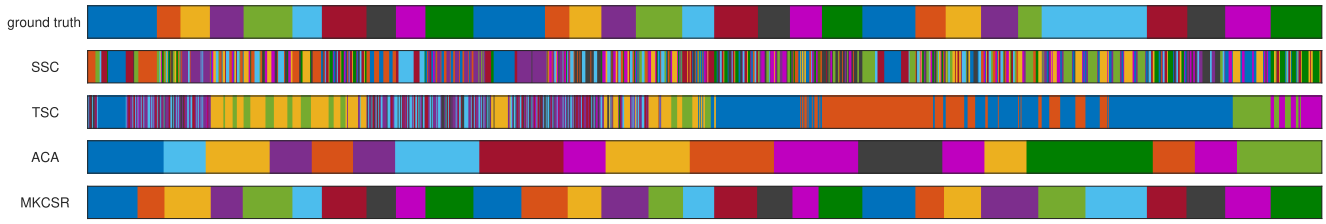
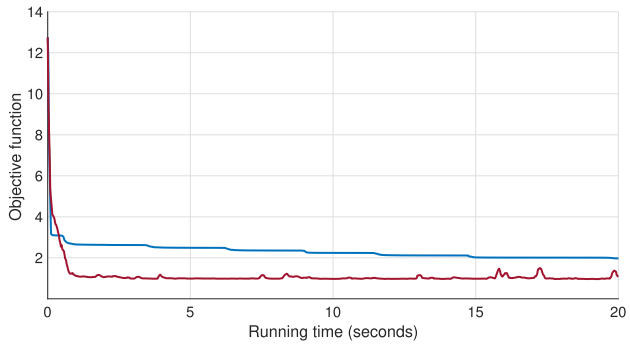
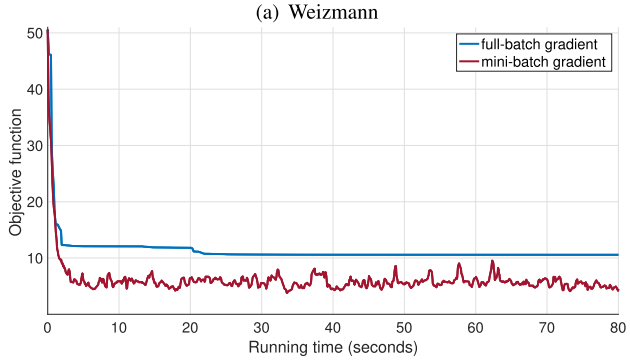


FIGURE 12. Visualization of segmentation results of SSC, TSC, ACA and MKCSR on three concatenated action video sequences from Weizmann dataset.



(a) Weizmann



(b) Google

FIGURE 13. Convergence curves of SKCSR (with stochastic gradients estimated from mini-batches $b = 256$) and KCSR (with gradients estimated from full batch (the whole data sequence)) on (a) Weizmann and (b) Google spoken digits datasets.

TABLE 3. Segmentation results on concatenated video sequences from Weizmann and MMI Facial AU datasets returned by different methods. The mean score of each methods over five random runs along with its variance are reported.

Dataset		SSC	TSC	ACA	MKCSR
Weizmann	ACC	0.1496 (0.0167)	0.1967 (0.0121)	0.5743 (0.0127)	**0.8509** (0.0139)
	NMI	0.1439 (0.0144)	0.2159 (0.0136)	0.5655 (0.0106)	**0.8732** (0.0150)
MMI Facial AU	ACC	0.2315 (0.0204)	0.3890 (0.0295)	0.6757 (0.0134)	**0.9351** (0.0103)
	NMI	0.2432 (0.0341)	0.3952 (0.0335)	0.6656 (0.0198)	**0.9221** (0.0095)

SSC, TSC and ACA. For all the compared methods, we set the number of clusters $k = 10$ and $k = 15$ for the Weizmann and MMI Facial AU data, respectively, and select the other parameters following the same scheme as mentioned in subsection V-D.

Fig. 12 visualizes the segmentation results on multiple video sequences from Weizmann data and Table 3 shows the evaluation scores on both Weizmann and MMI Facial AU datasets. Simultaneous segmentation of multiple data sequences is a challenging task. As we can observe that, in comparison with segmentation results of a single sequence (the second and third rows of Table 2), evaluation scores of SSC, TSC and ACA on the multiple data sequences are significantly reduced. MKCSR, however, compared to its original method KCSR, could preserve a great amount segmentation accuracy. As we can see that MKCSR obtained up to 0.8509 of ACC and 0.8732 of NMI on Weizmann data. For MMI Facial AU data, MKCSR also achieved 0.9351 of ACC and 0.9221 of NMI. These results validate that MKCSR can inherit advanced properties from SKCSR to perform efficiently and effectively on multiple data sequences.

VI. CONCLUSION

Approximation of segmentation for fast computational time and low memory requirement is very important as nowadays more and more large sequential datasets are available. Previous works for approximating optimal segmentation algorithm are either ineffective or inefficient because they still involve in optimization over discrete variables. In this paper, we proposed KCSR to alleviate the aforementioned issue. Our model combines a novel regularization based on sigmoid function with objective of balanced kernel $k-$means to approximate sequence segmentation. Its objective is differentiable almost every where. Therefore, we can use gradient-based algorithm to achieve the optimal segmentation. Note that, our model update all the parameters of interest in an unified manner. This is in contrast to existing approximation methods that sequentially update the segment boundaries, which has no guarantee on quality of the solutions. To further reduce the time and memory complexities, we introduce SKCSR – a stochastic variant of KCSR. SKCSR employs stochastic gradient descent, where the gradient is estimated from a randomly sampled subsequence, for updating parameters of the model. Thus, it can avoid storing large affinity and/or kernel matrix, which is a critical issue that inhibits existing methods from segmenting long data sequence. Finally, we modify the sigmoid-based regularization to develop MKCSR – an extended variant of KCSR for simultaneous segmentation of multiple data sequences. Through extensive experiments on various types of sequential data, performances of all the proposed models are evaluated and compared with those of existing methods.

The experimental results validates the claimed advantages of the proposed models.

APPENDIX
DERIVATION OF THE GRADIENT

In this section, we provide derivation of the gradient w.r.t $\boldsymbol{\gamma}$. Recall that our objective function is

$$J(\boldsymbol{\gamma}) = \operatorname{Tr}\left(\left(\boldsymbol{I}_n - \boldsymbol{G}^\top\left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1}\boldsymbol{G}\right)\boldsymbol{K}\right) + \lambda \operatorname{Tr}(\boldsymbol{G}\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{G}^\top). \tag{18}$$

The gradient $\nabla\boldsymbol{\gamma} = \frac{\partial J}{\partial \boldsymbol{\gamma}}$ can be computed using chain rule. We first compute the gradient of J w.r.t $\boldsymbol{G}$ as follows:

$$\frac{\partial J}{\partial \boldsymbol{G}} = 2\left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1}\boldsymbol{G}\boldsymbol{K}\boldsymbol{G}^\top\left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1}\boldsymbol{G} - 2\left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1}\boldsymbol{G}\boldsymbol{K} + \lambda\boldsymbol{G}\boldsymbol{1}\boldsymbol{1}^\top. \tag{19}$$

Since each entry in the j^{th} column of $\boldsymbol{G}$ is a function of continuously segment label τ_j we need to compute

$$\frac{\partial G_{i,j}}{\partial \tau_j} = \frac{\partial \max\left(0, 1 - |\tau_j - i|\right)}{\partial \tau_j} = \begin{cases} -1 & \text{if } i \le \tau_j \le i+1 \\ 1 & \text{if } i-1 \le \tau_j < i \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

Then the the gradient of J w.r.t $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_n]^\top$ is

$$\frac{\partial J}{\partial \tau_j} = \sum_{i=1}^{k} \frac{\partial J}{\partial G_{i,j}} \frac{\partial G_{i,j}}{\partial \tau_j}. \tag{21}$$

The segment label τ_j is again computed via a mixture of $k-1$ sigmoid functions, each of whose parameter is β_i. Thus, we need to compute

$$\frac{\partial \tau_j}{\partial \beta_i} = \frac{\partial\left(1 + \sum_{i'=1}^{k-1}\left(1 + e^{-\alpha(j-\beta_{i'})}\right)^{-1}\right)}{\partial \beta_i} = -\alpha\left(1 + e^{-\alpha(j-\beta_i)}\right)^{-1}\left[1 - \left(1 + e^{-\alpha(j-\beta_i)}\right)^{-1}\right]. \tag{22}$$

Then the gradient of J w.r.t $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{k-1}]^\top$ can be derived as follows

$$\frac{\partial J}{\partial \beta_i} = \sum_{j=1}^{n} \frac{\partial J}{\partial \tau_j} \frac{\partial \tau_j}{\partial \beta_i}. \tag{23}$$

Finally, we arrive at the gradient of J w.r.t $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_k]^\top$

$$\frac{\partial J}{\partial \gamma_c} = \sum_{i=1}^{k-1} \frac{\partial J}{\partial \beta_i} \frac{\partial \beta_i}{\partial \gamma_c}, \tag{24}$$

where

$$\frac{\partial \beta_i}{\partial \gamma_c} = \begin{cases} \dfrac{(n-1)e^{\gamma_c}}{\sum_{i'=1}^{k} e^{\gamma_{i'}}}\left(1 - \dfrac{\sum_{i'=1}^{i} e^{\gamma_{i'}}}{\sum_{i'=1}^{k} e^{\gamma_{i'}}}\right) & \text{if } c \le i, \\ -\dfrac{(n-1)e^{\gamma_c}\sum_{i'=1}^{i} e^{\gamma_{i'}}}{\left(\sum_{i'=1}^{k} e^{\gamma_{i'}}\right)^2} & \text{if } c > i. \end{cases} \tag{25}$$

REFERENCES

[1] W. Hu, S. Li, W. Zheng, Y. Lu, and G. Yu, "Robust sequential subspace clustering via ℓ_1-norm temporal graph," *Neurocomputing*, vol. 383, pp. 380–395, Mar. 2020.

[2] J. Zheng, P. Yang, G. Shen, S. Chen, and W. Zhang, "Enhanced low-rank constraint for temporal subspace clustering and its acceleration scheme," *Pattern Recognit.*, vol. 111, Mar. 2021, Art. no. 107678.

[3] T. Zhou, H. Fu, C. Gong, L. Shao, F. Porikli, H. Ling, and J. Shen, "Consistency and diversity induced human motion segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Feb. 1, 2022, doi: 10.1109/TPAMI.2022.3147841.

[4] Z. Harchaoui, F. Vallet, A. Lung-Yut-Fong, and O. Cappe, "A regularized kernel-based approach to unsupervised audio segmentation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 1665–1668.

[5] N. Seichepine, S. Essid, C. Fevotte, and O. Cappe, "Piecewise constant nonnegative matrix factorization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 6721–6725.

[6] A. E. Sakran, S. M. Abdou, S. E. Hamid, and M. Rashwan, "A review: Automatic speech segmentation," *Int. J. Comput. Sci. Mobile Comput.*, vol. 6, no. 4, pp. 308–315, 2017.

[7] M. Lavielle and G. Teyssiere, "Adaptive detection of multiple change-points in asset price volatility," in *Long Memory in Economics*. Berlin, Germany: Springer, 2007, pp. 129–156.

[8] Y.-W. Si and J. Yin, "OBST-based segmentation approach to financial time series," *Eng. Appl. Artif. Intell.*, vol. 26, no. 10, pp. 2581–2596, Nov. 2013.

[9] D. Hallac, P. Nystrup, and S. Boyd, "Greedy Gaussian segmentation of multivariate time series," *Adv. Data Anal. Classification*, vol. 13, no. 3, pp. 727–751, Sep. 2019.

[10] J. P. Vert and K. Bleakley, "Fast detection of multiple change-points shared by many signals using group LARS," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 2343–2351.

[11] R. Maidstone, T. Hocking, G. Rigaill, and P. Fearnhead, "On optimal multiple changepoint algorithms for large data," *Statist. Comput.*, vol. 27, no. 2, pp. 519–533, Mar. 2017.

[12] J. Reeves, J. Chen, X. L. Wang, R. Lund, and Q. Q. Lu, "A review and comparison of changepoint detection techniques for climate data," *J. Appl. Meteorol. Climatol.*, vol. 46, no. 6, pp. 900–915, Jun. 2007.

[13] J. Verbesselt, R. Hyndman, G. Newnham, and D. Culvenor, "Detecting trend and seasonal changes in satellite image time series," *Remote Sens. Environ.*, vol. 114, no. 1, pp. 106–115, Jan. 2010.

[14] S. Jamali, P. Jönsson, L. Eklundh, J. Ardö, and J. Seaquist, "Detecting changes in vegetation trends using time series segmentation," *Remote Sens. Environ.*, vol. 156, pp. 182–195, Jan. 2015.

[15] T. Heo and L. Manuel, "Greedy copula segmentation of multivariate non-stationary time series for climate change adaptation," *Prog. Disaster Sci.*, vol. 14, Apr. 2022, Art. no. 100221.

[16] C. Lévy-Leduc and F. Roueff, "Detection and localization of change-points in high-dimensional network traffic data," *Ann. Appl. Statist.*, vol. 3, no. 2, pp. 637–662, Jun. 2009.

[17] A. Lung-Yut-Fong, C. Lévy-Leduc, and O. Cappé, "Distributed detection/localization of change-points in high-dimensional network traffic data," *Statist. Comput.*, vol. 22, no. 2, pp. 485–496, Mar. 2012.

[18] Y. Song, P. Wu, D. Gilmore, and Q. Li, "A spatial heterogeneity-based segmentation model for analyzing road deterioration network data in multi-scale infrastructure systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 7073–7083, Nov. 2021.

[19] Z. Harchaoui and O. Cappe, "Retrospective mutiple change-point estimation with kernels," in *Proc. IEEE/SP 14th Workshop Stat. Signal Process.*, Aug. 2007, pp. 768–772.

[20] A. Celisse, G. Marot, M. Pierre-Jean, and G. J. Rigaill, "New efficient algorithms for multiple change-point detection with reproducing kernels," *Comput. Statist. Data Anal.*, vol. 128, pp. 200–220, Dec. 2018.

[21] C. Truong, L. Oudre, and N. Vayatis, "Greedy kernel change-point detection," *IEEE Trans. Signal Process.*, vol. 67, no. 24, pp. 6204–6214, Dec. 2019.

[22] S. Wright and J. Nocedal, *Numerical Optimization*. New York, NY, USA: Springer, 2006.

[23] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Process.*, vol. 167, Feb. 2020, Art. no. 107299.

[24] Z. Harchaoui, E. Moulines, and F. Bach, "Kernel change-point analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 609–616.

[25] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 723–773, 2012.
[26] S. Li, Y. Xie, H. Dai, and L. Song, "M-statistic for Kernel change-point detection," in *Proc. Adv. Neural Inf. Process. Syst.*, Montréal, QC, Canada, 2015, pp. 3366–3374.
[27] S. Li, Y. Xie, H. Dai, and L. Song, "Scan B-statistic for kernel change-point detection," *Sequential Anal.*, vol. 38, no. 4, pp. 503–544, Oct. 2019.
[28] S. Arlot, A. Celisse, and Z. Harchaoui, "A kernel multiple change-point algorithm via model selection," *J. Mach. Learn. Res.*, vol. 20, no. 162, pp. 1–56, 2019.
[29] M. Hoai and F. De la Torre, "Maximum margin temporal clustering," in *Artificial Intelligence and Statistics*. PMLR, 2012, pp. 520–528.
[30] F. Zhou, F. De la Torre, and J. K. Hodgins, "Hierarchical aligned cluster analysis for temporal clustering of human motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 3, pp. 582–596, Mar. 2013.
[31] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[32] S. Tierney, J. Gao, and Y. Guo, "Subspace clustering for sequential data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1019–1026.
[33] F. Wu, Y. Hu, J. Gao, Y. Sun, and B. Yin, "Ordered subspace clustering with block-diagonal priors," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 3209–3219, Dec. 2016.
[34] 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.
[35] H. Liu, J. Cheng, and F. Wang, "Sequential subspace clustering via temporal smoothness for sequential data segmentation," *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 866–878, Feb. 2018.
[36] L. Clopton, E. Mavroudi, M. Tsakiris, H. Ali, and R. Vidal, "Temporal subspace clustering for unsupervised action segmentation," CSMR REU, 2017, pp. 1–7.
[37] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel K-means: Spectral clustering and normalized cuts," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2004, pp. 551–556.
[38] F. De la Torre, "A least-squares framework for component analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1041–1055, Jun. 2012.
[39] R Zass and A Shashua, "A unifying approach to hard and probabilistic clustering," in *Proc. 10th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 1, Oct. 2005, pp. 294–301.
[40] S. Zhong and J. Ghosh, "Model-based clustering with soft balancing," in *Proc. 3rd IEEE Int. Conf. Data Mining*, Nov. 2003, p. 459.
[41] H. Liu, J. Han, F. Nie, and X. Li, "Balanced clustering with least square regression," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017, pp. 1–7.
[42] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific J. Math.*, vol. 16, no. 1, pp. 1–3, Jan. 1966.
[43] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, Sep. 1951.
[44] L. Bottou, "Online learning and stochastic approximations," *On-Line Learn. neural Netw.*, vol. 17, no. 9, p. 142, 1998.
[45] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. vol. 65. Hoboken, NJ, USA: Wiley, 2005.
[46] Z. Allen-Zhu, "Katyusha: The first direct acceleration of stochastic gradient methods," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 8194–8244, 2017.
[47] M. Schmidt, N. L. Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Math. Program.*, vol. 162, nos. 1–2, pp. 83–112, Mar. 2017.
[48] J. Qiu, X. Wang, P. Fua, and D. Tao, "Matching seqlets: An unsupervised approach for locality preserving sequence matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 745–752, Feb. 2019.
[49] C.-Y. Chang, D.-A. Huang, Y. Sui, L. Fei-Fei, and J. C. Niebles, "D3TW: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3546–3555.
[50] J. Li and S. Todorovic, "Set-constrained Viterbi for set-supervised action segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10820–10829.
[51] H. Shimodaira, K. Noma, M. Nakai, and S. Sagayama, "Dynamic time-alignment kernel in support vector machine," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, 2001, pp. 921–928.
[52] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 12, pp. 2247–2253, Dec. 2007.
[53] M. Hoai and F. De la Torre, "Max-margin early event detectors," *Int. J. Comput. Vis.*, vol. 107, no. 2, pp. 191–202, Apr. 2014.
[54] M. Pantic, M. Valstar, R. Rademaker, and L. Maat, "Web-based database for facial expression analysis," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2005, p. 5.
[55] D. P. Tung and A. Takasu, "Deep multiview learning from sequentially unaligned data," *IEEE Access*, vol. 8, pp. 217928–217946, 2020.
[56] D. E. King, "Dlib-ml: A machine learning toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, Dec. 2009.
[57] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018, *arXiv:1804.03209*.
[58] B. McMahan and D. Rao, "Listening to the world improves speech command recognition," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 1–8.
[59] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-28, no. 4, pp. 357–366, Aug. 1980.
[60] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 11, no. 86, pp. 2278–2324, Nov. 1998.
[61] P. Casale, O. Pujol, and P. Radeva, "Personalization and user verification in wearable systems using biometric walking patterns," *Pers. Ubiquitous Comput.*, vol. 16, no. 5, pp. 563–580, Jun. 2012.
[62] D. Cai, X. He, and J. Han, "Document clustering using locality preserving indexing," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 12, pp. 1624–1637, Dec. 2005.
[63] L. Lovasz and M. Plummer, *Matching Theory*. North Holland, Budapest: Akademiai Kiado, 1986.

TUNG DOAN received the B.S. degree in computer engineering from the Hanoi University of Science and Technology, in 2014, and the Ph.D. degree from the National Institute of Informatics, Japan, in 2021. He is currently a Staff Lecturer at the Department of Computer Engineering, School of Information and Communication Technology, Hanoi University of Science and Technology. His current research interests include deep learning, multiview learning, generative model, and sequential data.

ATSUHIRO TAKASU (Member, IEEE) received the B.E., M.E., and Dr.Eng. degrees from The University of Tokyo, Japan, in 1984, 1986, and 1989, respectively. He is currently a Professor at the National Institute of Informatics, Japan. His research interests include data engineering and data mining. He is a member of the ACM, IEICE, IPSJ, and JSAI.