Kernel Clustering with Sigmoid Regularization for Efficient Segmentation of Sequential Data

TUNG DOAN and ATSUHIRO TAKASU (Member, IEEE)

1 Hanoi University of Science and Technology, No. 1, Dai Co Viet road, Hanoi, Vietnam
2 The Graduate University for Advanced Studies, SOKENDAI, Kanagawa 240-0193 Japan
3 National Institute of Informatics, Tokyo 101-8430, Japan

Corresponding author: Tung Doan (e-mail: 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 non-linear 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 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 $O(n^2)$ 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 IV we review related works that perform segmentation based on clustering methods. Next, we briefly presents some background for our proposed models in Section V. Section VI 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 VII we conclude the paper in Section VIII.

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 [24]–[28] 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 are 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], [29] employed dynamic programming (DP) algorithm to optimally obtain the segment boundaries. However, because DP have time complexity of order $O(n^2)$ (including computational time of the cost matrix

$O(n \log n)$
in the feature space), it is impractical for handling long data sequences. To reduce the time complexity, \cite{21} proposed a greedy algorithm that sequentially detects change points one at an iteration. \cite{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 \cite{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) \cite{30} and Aligned clustering analysis (ACA) \cite{31} divide data sequences into a set of non-overlapping short segments. These subsequences are then partitioned into \( k \) classes using unsupervised support vector machine \cite{32} or kernel \( k \)-means clustering \cite{33}. Recently, a branch of methods based on subspace clustering has been proposed. These methods often include two steps. First, given a data sequences \( X = [x_1, \ldots, x_n] \), they learn a new representation (coding matrix) \( Z = [z_1, \ldots, z_n] \) that characterizes the underlying subspace structures and sequential (a.k.a. temporal) information of the original data. Second, the normalized cut algorithm (Ncut) \cite{34} is then utilized for segmentation of \( Z \).

To preserve the sequential information in the new representation, \cite{35,36} proposed a linear regularization of the form \( ||Z^TA||_{1,2} \), where \( R \in \mathbb{R}^{n \times (n-1)} \) 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 \( z_j \) and \( z_{j+1} \) of the two consecutive samples \( x_j \) and \( x_{j+1} \), respectively, are forced to be similar. \cite{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, \cite{37,38} proposed manifold-based regularization that preserves the sequential information for the local neighborhood data samples. This type of regularization is more preferable \cite{2} as it often outperforms the linear one in most tests \cite{39}. 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 CD algorithms 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 \cite{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 III.

![FIGURE 1. An example of sequence segmentation: (top) an example sequence of length 23 and (bottom) the corresponding indicator matrix with number of segments \( k = 7 \).](image1.png)

**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 \( A \), its \( i^{th} \) column is denoted as \( a_i \) and its element at position \((j, i)\) is expressed by \( a_{j,i} \) or \( A_{j,i} \). The transpose matrix of \( A \) is denoted by \( A^\top \). If \( A \) is a square matrix of size \( n \) then its trace is expressed as \( \text{Tr}(A) = \sum_{i=1}^{n} A_{i,i} \). If \( A \in \{0,1\}^{k \times n} \) then for any given element \( A_{j,i} \), we have \( A_{j,i} = 0 \) or \( A_{j,i} = 1 \) (\( A \) 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 \( X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n} \) denotes the given sequence of length \( n \) and dimension \( d \). 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 \( G \in \{0,1\}^{k \times n} \), whose each element is as follows

\[
G_{i,j} = \begin{cases} 
1 & x_j \in \text{segment } i, \\
0 & \text{otherwise.} 
\end{cases}
\]
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 \(X\) 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 \(\phi(x_i) = \kappa(x_i, \cdot)\), resulting the inner-product \(\phi(x_i)\phi(x_j) = \kappa(x_i, x_j)\). 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

\[
\arg\min_{G \in \mathcal{G}} \sum_{j=1}^{n} \sum_{i=1}^{k} G_{j,i} \left\| \phi(x_i) - \mu_j \right\|^2, \tag{2}
\]

where \(\mathcal{G}\) is the set of all valid sample-to-segment indicator matrices and \(\mu_j\) 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 \(G\) 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 \(K \in \mathbb{R}^{n \times n}\), where each element \(K_{i,j} = \phi(x_i)\phi(x_j) = \kappa(x_i, x_j)\) represents how likely the two samples are assigned to the same class. Let \(G \in \{0, 1\}^{k \times n}\) denotes the associated (unknown) sample-to-class indicator matrix of \(X\), where \(G_{i,j} = 1\) if \(x_j\) is assigned to the \(i^{th}\) class and zero otherwise. Here, different from the segmentation task, there is no constraint on the indicator matrix \(G\). Then the objective function of kernel \(k\)-means [38]–[40] can be expressed as follows:

\[
J_{KKM}(G) = \text{Tr}(LK) , \tag{3}
\]

where \(L = I_n - GG^\top (GG^\top)^{-1} G\).

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 [41]. To alleviate this issue, recently [42] has proposed a simple regularization on the indicator matrix of the form \(\text{Tr}(GG^\top G^\top)\), where \(1\) 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 [41] and the regularization to form a new objective of balanced kernel \(k\)-means

\[
J_{BKKM}(G) = \text{Tr}(LK) + \lambda \text{Tr}(GG^\top G^\top) , \tag{4}
\]

where \(\lambda\) is a positive parameter that controls the balanced regularization.

**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 [41] for segmentation of data sequence \(X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}\). 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 \(\beta\) specifies the midpoint and \(\alpha\) controls the steepness of the function curve at the midpoint. Figure 5 depicts a sigmoid function, where the midpoint \(\beta\) is fixed at 11.5 and the parameter \(\alpha\) varies from 0.1 to 10.
We can observe that the higher \( \alpha \) 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 \( \beta \) approximates the segment boundary. If we denote \( \tau_j \in [1, 2] \) (continuously valued) as segment label of sample \( x_j \), then

\[
\tau_j \approx 1 + f_{\text{sigmoid}}(j, \alpha, \beta).
\]

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 \( \beta_i \) for \( 1 \leq i < k - 1 \).

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

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

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

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, \( \beta_i \) always satisfies \( 1 \leq \beta_i \leq n \). In addition, the ratio becomes larger as \( i \) increases. This guarantees that \( \beta_{i'} < \beta_i \) for \( 1 \leq i' < i \leq k - 1 \).

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

\[
G_{i,j} \approx \max \{0, 1 - |\tau_j - i|\}.
\]

This equation maps the segment label \( \tau_j \) from the range \([1, k]\) to the range \([0, 1]\) for approximating the sample-to-segment indicator matrix.

**Algorithm 1:** Gradient descent algorithm for KCSR

**Input:** Kernel matrix \( K \), number of segments \( k \), steepness parameter \( \alpha \), tolerance \( \epsilon \).

**Output:** Optimal parameters \( \gamma^* = [\gamma_1^*, \ldots, \gamma_k^*]^T \).

1: repeat
2: compute gradient \( \nabla J(\gamma) = \frac{\partial J(\gamma)}{\partial \gamma} \)
3: compute stepsizes \( \eta \) using Armijo-Goldstein line search [22, 43];
4: update \( \gamma(t+1) = \gamma(t) - \eta \nabla J(\gamma(t)) \);
5: until \( |J(\gamma(t+1)) - J(\gamma(t))| \leq \epsilon \)

We now can formulate an optimization problem that combines objective of the balanced kernel clustering with sigmoid-based regularization for segmentation. Let \( K \in \mathbb{R}^{n \times n} \) be the kernel matrix of the data sequence \( X \) then our kernel-based segmentation optimization problem is

\[
\arg\min_{\gamma_1, \ldots, \gamma_k} \text{Tr}(LK) + \lambda \text{Tr}(G11^T G^T)
\]

s.t. \( L = I_n - G^T (GG^T)^{-1} G \),

\[
G_{i,j} = \max \{0, 1 - |\tau_j - i|\} \quad \forall i, j,
\]

\[
\tau_j = 1 + \sum_{i=1}^{k-1} f_{\text{sigmoid}}(j, \alpha, \beta_i) \quad \forall j,
\]

\[
\beta_i = \left( 1 - \frac{\sum_{i'=1}^{i} e^{\gamma_{i'}}}{\sum_{i'=1}^{k} e^{\gamma_{i'}}} \right) + n \times \frac{\sum_{i'=1}^{i} e^{\gamma_{i'}}}{\sum_{i'=1}^{k} e^{\gamma_{i'}}} \quad \forall i.
\]

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

\[
\nabla J = \frac{\partial J(\gamma)}{\partial \gamma} = \frac{\partial J(\gamma)}{\partial G} \times \frac{\partial G}{\partial \tau} \times \frac{\partial \tau}{\partial \beta} \times \frac{\partial \beta}{\partial \gamma},
\]

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

**FIGURE 3.** Sigmoid function with different values of the parameter \( \alpha \).

**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 \( k = 7 \).
B. STOCHASTIC KCSR

Kernel segmentation allow us to capture nonlinear structure in the data. However, this advantage is achieved at the expense 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 \( K \), 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 125\( K \). The corresponding kernel matrix \( K \) 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)\(^1\). 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 \( X(t) \), where \( t \) expresses the iteration index. Then, the stochastic gradient is estimated as follows:

\[
\nabla \gamma = \frac{\partial J(\gamma)}{\partial G(t)} + \frac{\partial G(t)}{\partial \tau} \frac{\partial \tau}{\partial \beta} \frac{\partial \beta}{\partial \gamma}.
\]

In equation \( \frac{\partial J(\gamma)}{\partial G(t)} \) is only associated with a partial kernel matrix \( K(t) \in \mathbb{R}^{b \times b} \), which corresponds to the samples in \( X(t) \). Therefore, it is much more efficient in terms of both running time and memory consumption than computing the full-batch gradient as in equation \([12]\). Details of the algorithm is given in Algorithm \( \text{2} \) and complexity comparison between the proposed methods and several baselines are given in Table \( \text{1} \). We note that convergences of both gradient with step size found by Armijo-Goldstein line search\(^2\) and stochastic gradient descent algorithms with vanishing step size are theoretically proven. In fact, it is well-known\(^4\) 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.

### TABLE 1. Time and space complexities of different segmentation methods.

| Method   | SSC | TSC | ACA | AKS | GKS | KCSR | SKCSR |
|----------|-----|-----|-----|-----|-----|------|-------|
| Time     | \( O(n^2 + n^2) \) | \( O(n^2 + n^2) \) | \( O(n^2 + n^2) \) | \( n(n + r \log(k)n) \) | \( O(k(n + n^2)) \) | \( O(n^2) \) | \( O(n(bk^2 + b^2)) \) |
| Space    | \( O(n^2) \) | \( O(n^2) \) | \( O(n^2 + n^2) \) | \( O((k + r)n) \) | \( O(n^2) \) | \( O(n^2) \) | \( O(b^2) \) |

\( k \) denotes the maximum length of divided subsequences in ACA, \( d \) is dimension of the new representation \( Z \) in OSC and TSC. \( r \) 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 \( b \ll n \).

### Algorithm 2: Stochastic gradient descent algorithm for KCSR

**Input:** Data sequence \( X \), number of segments \( k \), steepness parameter \( \alpha \), number of iterations \( T \), minibatch size \( b \), initial learning rate \( \eta_0 \), momentum \( \mu \in [0, 1] \), weight decay \( \rho \in (0, 1] \).

**Output:** Optimal parameters \( \gamma = [\gamma_1, \ldots, \gamma_k]^T \).

1. for \( t = 1, \ldots, T \) do
   2. \( \eta = \eta_0 \times \rho^t \);
   3. randomly sample a sub-sequence \( X(t) \) of length \( b \);
   4. compute the partial kernel matrix \( K(t) \);
   5. compute the stochastic gradient \( \nabla \gamma = \frac{\partial J}{\partial G} \) based on \( K(t) \) and original indexes of samples in \( X(t) \);
   6. \( \Delta \gamma(t) = \eta \nabla \gamma - \mu \Delta \gamma(t-1) \);
   7. \( \gamma(t) = \gamma(t-1) + \Delta \gamma(t) \);
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\(^{49-51}\).

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 \( m \) sequences
are matched \( \mathcal{M} \). Let \( X_p \in \mathbb{R}^{d \times n_p} \) for \( 1 \leq p \leq m \) denote the \( p \)-th data sequence and \( G_p \in \mathbb{R}^{k \times n_p} \) be its corresponding sample-to-segment indicator matrix. MKSSR firstly concatenates all the sequences to form a single long sequence \( X = [x_1, \ldots, x_m] \in \mathbb{R}^{d \times n} \), where \( n = \sum_{i=1}^{m} n_p \). Then \( G = [G_1, \ldots, G_m] \in \mathbb{R}^{k \times n} \) is the corresponding indicator matrix of \( X \). Similar to the original KCSR, each element of \( G \) is defined as in (7). However, in MKCSR, the segment label \( \tau_j \) is computed as following

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

(13)

The function (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 \( \gamma_1, \ldots, \gamma_{mk} \).  

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 depicted in the two first subfigures, can smoothly approximate the indicator matrix in the toy example.
such that for \((p-1)(k-1) + 1 \leq j \leq p(k-1)\)

\[
\beta_j = \left(1 + \sum_{q=1}^{p-1} n_q \right) \left(1 - \frac{\sum_{i=(p-1)k+1}^{pk} e^{\gamma_{i,j}}}{\sum_{i=(p-1)k+1}^{pk} e^{\gamma_{i,j}}} \right) + \sum_{q=1}^{p} n_q \frac{\sum_{i=(p-1)k+1}^{pk} e^{\gamma_{i,j}}}{\sum_{i=(p-1)k+1}^{pk} e^{\gamma_{i,j}}}
\]

(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)** \([31]\) – a temporal clustering method that combines \(k\)-means with Dynamic time alignment kernel \([32]\).
- **Sequential subspace clustering (SSC)** \([1]\) – a temporal clustering method that combines subspace clustering with linearly temporal regularization weighted by \(\ell_1\)-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.

- **Synthetic data.** We first generate 2D data samples that form four circles of different diameters. They are illustrated in Figure 8. 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.

- **Weizmann data.** The Weizmann dataset \([53]\) 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 \([34]\), 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 x 55. For each 70-by-55 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.

**MMI Facial action units.** We exploit the MMI Facial Expression dataset \([55]\), 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 \([57]\). 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 labels are already given in the original dataset.

**Google spoken digits.** Google’s Speech Commands (GSC) \([58], [59]\) 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) \([60]\) 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.
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.

**Ordered MNIST data.** The MNIST dataset [61] 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 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.

**Acceleration data.** The acceleration data [62] 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 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 acceleration dataset is available in UCI repositories and can be retrieved from [https://archive.ics.uci.edu/ml/machine-learning-databases/00287/](https://archive.ics.uci.edu/ml/machine-learning-databases/00287/).
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.

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 discontinuous segments. Since, all the compared methods base on clustering scheme, we use accuracy and normalized mutual information [3] as evaluation metrics to assess the segmentation results.

Let $\mathcal{L} = [l_1, \ldots, l_n]$ and $\mathcal{C} = [c_1, \ldots, c_n]$ be the obtained labels and ground-truth labels of a given data sequence $X = [x_1, \ldots, x_n]$. $l_j = i$ (similar for $c_i$) for $1 \leq i \leq k$ indicates that $x_j$ belongs to cluster (segment) $c_i$. The accuracy (ACC) is defined as follows:

$$ACC = \frac{\sum_{j=1}^{n} \delta(l_j, \text{map}(l_j))}{n},$$  

(15)

where $\delta(a,b)$ is the delta function that equals one if $a = b$ and zero otherwise and $\text{map}(l_j)$ is the permutation mapping function that maps label $l_j$ to the equivalent ground truth label. In this work, we use Kuhn-Munkres algorithm [64] to find the mapping.

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

$$MI(\mathcal{C}, \hat{\mathcal{C}}) = \sum_{c_i \in \mathcal{C}, \hat{c}_i \in \hat{\mathcal{C}}} p(c_i, \hat{c}_i) \log_2 \frac{p(c_i, \hat{c}_i)}{p(c_i)p(\hat{c}_i)},$$  

(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

FIGURE 8. Synthetic experiment: (a) data generated in 2D space, (b) the data after contiguously rearranging and visualization of segmentation results returned by all the compared methods. Different colors represent different clusters.
that the selected data sample belongs to both $c_i$ and $c_{i'}$. This metric is normalized to the range $[0, 1]$ as follows:

$$NMI(C, \hat{C}) = \frac{MI(C, \hat{C})}{\max(H(C), H(\hat{C}))},$$

(17)

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

### 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 $Z$. 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 $\epsilon$ for convergence verification in KCSR is fixed at $10^{-6}$. For all the datasets, we use the Radial Basis Function (RBF) Kernel $^3$ with proper width $\sigma$ 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. However, different from SSC and TSC, where the regularization is just local, 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 noisy 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 $\beta$ of the

---

3 RBF kernel: $\kappa(x_i, x_j) = e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}$. 

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

---

| Dataset          | ACC | TSC    | ACA    | AKS    | GKS    | KCSR   | SKCSR   |
|------------------|-----|--------|--------|--------|--------|--------|---------|
| Synthetic data   | 0.9695 (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) |
| Weizmann         | 0.1070 (0.0649) | 0.3068 (0.0781) | 0.9214 (0.0375) | 0.6967 (0.0791) | 0.7036 (0.0325) | 0.9959 (0.0024) | 0.9847 (0.0074) |
| MMI Facial AU    | 0.4566 (0.0289) | 0.7028 (0.0430) | 0.7687 (0.0140) | 0.7182 (0.0185) | 0.5628 (0.0219) | 0.8885 (0.0089) | 0.8964 (0.0113) |
| Google           | 0.6153 (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) |
| Ordered MNIST    | 0.6014 (0.0112) | 0.6321 (0.0437) | 0.7502 (0.0205) | 0.7600 (0.0241) | 0.6355 (0.0258) | 0.9625 (0.0129) | 0.9680 (0.0126) |
| Acceleration     | 0.1589 (0.0229) | 0.6314 (0.0409) | 0.8241 (0.0182) | 0.6260 (0.0257) | 0.7458 (0.0294) | 0.7914 (0.0112) | 0.8380 (0.0105) |

---

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
summation of sigmoid functions. However, different from these heuristic approximations that search for the segment boundaries sequentially, the proposed models simultaneously obtain all the $\beta$ via gradient-based algorithm. As convergence of this optimization algorithm is theoretically proved, optimality of the solutions is guaranteed. To qualitatively assess the 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 $\gamma$ 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 [45]. 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 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 VI.

Fig. 12 visualizes the segmentation results on multiple video sequences from 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 of 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 de-
velop 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 validate the claimed advantages of the proposed models.

**APPENDIX A DERIVATION OF THE GRADIENT**

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

$$J(\gamma) = \text{Tr} \left( \left( \begin{bmatrix} 1 & -G \end{bmatrix} \right) \left( GG^T \right)^{-1} G \right) K + \lambda \text{Tr}(G\Gamma_1^T G^\top) \quad \text{(18)}$$

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

$$\frac{\partial J}{\partial G} = 2 \left( GG^T \right)^{-1} G K G^T \left( GG^T \right)^{-1} G - 2 \left( GG^T \right)^{-1} G K + \lambda G\Gamma_1^T \quad \text{(19)}$$

Since each entry in the $j$th column of $G$ is a function of continuously segment label $\tau_j$ we need to compute

$$\frac{\partial G_{i,j}}{\partial \tau_j} = \max \left( 0, 1 - |\tau_j - i| \right) \quad \text{(20)}$$

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

$$\frac{\partial J}{\partial \tau_j} = \sum_{i=1}^{k-1} \frac{\partial J}{\partial G_{i,j}} \frac{\partial G_{i,j}}{\partial \tau_j} \quad \text{(21)}$$

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

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

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

$$\frac{\partial J}{\partial \beta_i} = \sum_{j=1}^{n} \frac{\partial J}{\partial \tau_j} \frac{\partial \tau_j}{\partial \beta_i} \quad \text{(23)}$$

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

$$\frac{\partial J}{\partial \gamma_c} = \sum_{i=1}^{k-1} \frac{\partial J}{\partial \beta_i} \frac{\partial \beta_i}{\partial \gamma_c} \quad \text{(24)}$$

where

$$\frac{\partial \beta_i}{\partial \gamma_c} = \begin{cases} \frac{(n-1) e^{\gamma_i} \sum_{j=1}^{n} e^{\gamma_j} - \sum_{j=1}^{n} e^{\gamma_j} e^{\gamma_c}}{\left( \sum_{j=1}^{n} e^{\gamma_j} \right)^2} & \text{if } c \leq i, \\ \frac{-\sum_{j=1}^{n} e^{\gamma_j} e^{\gamma_c}}{\left( \sum_{j=1}^{n} e^{\gamma_j} \right)^2} & \text{if } c > i. \end{cases} \quad \text{(25)}$$

**REFERENCES**

[1] Hu, Wenyu, Shenghao Li, Weidong Zheng, Yao Lu, and Gaohang Yu. “Robust sequential subspace clustering via $\ell_1$-norm temporal graph.” Neurocomputing 383 (2020): 380–395.

[2] Zheng, Jianwei, Ping Yang, Guojian Shen, Shengyong Chen, and Wei Zhang. “Enhanced low-rank constraint for temporal subspace clustering and its acceleration scheme.” Pattern Recognition 111 (2021): 107678.

[3] Zhou, Tao, Huazhu Fu, Chen Gong, Ling Shao, Fatih Porikli, Haibing Lin, and Jianheng Shen. “Consistency and Diversity induced Human Motion Segmentation.” IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

[4] Harchaoui, Zaid, Félixien Vallet, Alexandre Lung-Yut-Fong, and Olivier Cappé. “A regularization kernel-based approach to unsupervised audio segmentation.” In 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1665–1668. IEEE, 2009.

[5] Seichepine, Nicolas, Slim Essid, Cédric Févotte, and Olivier Cappé. “Piecewise constant nonnegative matrix factorization.” In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6721–6725. IEEE, 2014.

[6] Sakran, Alaa Elhab, Sherif Mahdy Abdou, Salah Eldeen Hamid, and Mohsen Rashwan. “A review: Automatic speech segmentation.” International Journal of Computer Science and Mobile Computing 6, no. 4 (2017): 308–315.

[7] Lavielle, Marc, and Gilles Teyssière. “Adaptive detection of multiple change-points in asset price volatility.” In Long memory in economics, pp. 129–156. Springer, Berlin, Heidelberg, 2007.

[8] Si, Yain-Whur, and Jiaying Yin. “OBSF-based segmentation approach to financial time series.” Engineering Applications of Artificial Intelligence 26, no. 10 (2013): 2581–2596.

[9] Hallac, David, Peter Nystrup, and Stephen Boyd. “Greedy Gaussian segmentation of multivariate time series.” Advances in Data Analysis and Classification 13, no. 3 (2019): 727–751.

[10] Vert, Jean-Philippe, and Kevin Bleakley. “Fast detection of multiple change-points shared by many signals using group LARS.” Advances in neural information processing systems 23 (2010): 2343–2351.

[11] Maidstone, Robert, Toby Hocking, Guillem Rigall, and Paul Fearnhead. “On optimal multiple changepoint algorithms for large data.” Statistics and computing 27, no. 2 (2017): 519–533.

[12] Reeves, Jack, Jien Chen, Xiaolan L. Wang, Robert Lund, and Qi Qi Lu. “A review and comparison of changepoint detection techniques for climate data.” Journal of applied meteorology and climatology 46, no. 6 (2007): 900–915.

[13] Betsbex, Jan, Rob Hyndman, Glenn Newham, and Dariusz Culenov. “Detecting trend and seasonal changes in satellite image time series.” Remote sensing of Environment 114, no. 1 (2010): 106–115.

[14] Jamali, Sadegh, Per Jönsson, Lars Ekklundh, Jonas Aråd, and Jonathan Seaqust. “Detecting changes in vegetation trends using time series segmentation.” Remote Sensing of Environment 156 (2015): 182–195.

[15] Heo, Taemin, and Lance Manuel. “Greedy copula segmentation of multivariate non-stationary time series for climate change adaptation.” Progress in Disaster Science (2022): 100221.

[16] Lévy-Leduc, Céline, and François Roueff. “Detection and localization of change-points in high-dimensional network traffic data.” Annals of Applied Statistics 3, no. 2 (2009): 637–662.

[17] Lung-Yut-Fong, Alexandre, Céline Lévy-Leduc, and Olivier Cappé. “Distributed detection/localization of change-points in high-dimensional network traffic data.” Statistics and Computing 22, no. 2 (2012): 485–496.

[18] Song, Yongze, Peng Wu, Daniel Gilmore, and Qiongdong Li. “A spatial heterogeneity-based segmentation model for analyzing road deterioration network data in multi-scale infrastructure systems.” IEEE Transactions on Intelligent Transportation Systems 22, no. 11 (2020): 7073–7083.

[19] Harchaoui, Zaid, and Olivier Cappé. “Retrospective multiple change-point estimation with kernels.” In 2007 IEEE/SP 14th Workshop on Statistical Signal Processing, pp. 768–772. IEEE, 2007.
[20] Celisse, Alain, Guillelmette Marot, Morgane Pierre-Jean, and G. J. Rigaill. “New efficient algorithms for multiple change-point detection with reproducing kernels.” Computational Statistics & Data Analysis 128 (2018): 200–220.

[21] Truong, Charles, Laurent Oudre, and Nicolas Vayatis. “Greedy kernel change-point detection.” IEEE Transactions on Signal Processing 67, no. 24 (2019): 6204–6214.

[22] Nocedal, Jorge, and Stephen Wright. Numerical optimization. Springer Science & Business Media, 2006.

[23] Truong, Charles, Laurent Oudre, and Nicolas Vayatis. “Selective review of offline change point detection methods.” Signal Processing 167 (2020): 107299.

[24] Harchaoui, Zaid, Eric Moulines, and Francis R. Bach. “Kernel change-point analysis.” In Advances in neural information processing systems, pp. 609–616. 2009.

[25] Harchaoui, Zaid, Félicien Vallet, Alexandre Lung-Yut-Fong, and Olivier Cappe. “A regularized kernel-based approach to unsupervised audio segmentation.” In 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1665–1668. IEEE, 2009.

[26] Grettou, Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. “A kernel two-sample test.” The Journal of Machine Learning Research 13, no. 1 (2012): 723–773.

[27] Li, Shuang, Yao Xie, Hanjun Dai, and Le Song. “M-statistic for Kernel change-point detection.” in Proc. Adv. Neural Inf. Process. Syst., Montréal, QC, Canada, 2015, pp. 3366–3374.

[28] Li, Shuang, Yao Xie, Hanjun Dai, and Le Song. “Scan B-statistic for kernel change-point detection.” Sequential Analysis 38, no. 4 (2019): 503–544.

[29] Arlot, Sylvain, Alain Celisse, and Zaid Harchaoui. “A kernel multiple change-point algorithm via model selection.” Journal of machine learning research 20, no. 162 (2019).

[30] Hou, Minh, and Fernando De la Torre. “Maximum margin temporal clustering.” In Artificial Intelligence and Statistics, pp. 520–528. PMLR, 2012.

[31] Zhou, Feng, Fernando De la Torre, and Jessica K. Hodiggins. “Hierarchical Aligned Cluster Analysis for Temporal Clustering of Human Motion.” IEEE Transactions on Pattern Analysis and Machine Intelligence 35, no. 3 (2013): 582–596.

[32] Shi, Jianbo, and Jitendra Malik. “Normalized cuts and image segmentation.” IEEE Transactions on pattern analysis and machine intelligence 22, no. 8 (2000): 888–905.

[33] Tierney, Stephen, Junbin Gao, and Yi Guo. “Subspace clustering for sequential data.” In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1019–1026. 2014.

[34] Wu, Fei, Yongchi Hu, Junbin Gao, Yanfeng Sun, and Baoqai Yin. “Ordered Subspace Clustering With Block-Diagonal Priors.” IEEE transactions on cybernetics 46, no. 12 (2016): 3209–3219.

[35] Li, Sheng, Kang Li, and Yun Fu. “Temporal subspace clustering for human motion segmentation.” In Proceedings of the IEEE International Conference on Computer Vision, pp. 4453–4461. 2015.

[36] Liu, Hanjun, Jian Cheng, and Feng Wang. “Sequential subspace clustering via temporal smoothness for sequential data segmentation.” IEEE Transactions on Image Processing 27, no. 2 (2017): 866–878.

[37] Clotpon, Larissa, Efrosyni Mavroudi, Manolis Tsaikiris, Haider Ali, and René Vidal. “Temporal subspace clustering for unsupervised action segmentation.” CSMR REU (2017): 1–7.

[38] Dhillon, Inderjit S., Yuqiang Guan, and Brian Kulis. “Kernel k-means: spectral clustering and normalized cuts.” In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 551–556. 2004.

[39] De la Torre, Fernando. “A least-squares framework for component analysis.” IEEE Transactions on Pattern Analysis and Machine Intelligence 34, no. 6 (2012): 1041–1055.

[40] Zass, Ron, and Amnon Shashua. “A unifying approach to hard and probabilistic clustering.” In Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1, vol. 1, pp. 294–301. IEEE, 2005.

[41] Zhong, Shi, and Joydeep Ghosh. “Model-based clustering with soft balancing.” In ICDM’03: Proceedings of the Third IEEE International Conference on Data Mining, p. 459. 2003.

[42] Liu, Huayang, Junwei Han, Feiping Nie, and Xuelong Li. “Balanced clustering with least square regression.” In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, no. 1. 2017.

[43] Armijo, Larry. “Minimization of functions having Lipschitz continuous first partial derivatives.” Pacific Journal of mathematics 16, no. 1 (1966): 1–3.

[44] Robbins, Herbert, and Sutton Monro. “A stochastic approximation method.” The annals of mathematical statistics (1951): 400–407.

[45] Bottou, Léon. “Online learning and stochastic approximations.” On-line learning in neural networks 17, no. 9 (1998): 142.

[46] Spall, James C. Introduction to stochastic search and optimization: estimation, simulation, and control. Vol. 65. John Wiley & Sons, 2005.

[47] Allen-Zhu, Zeyuan. “ Kathyusha: The first direct acceleration of stochastic gradient methods.” The Journal of Machine Learning Research 18, no. 1 (2017): 8194–8244.

[48] Schmidt, Mark, Nicolas Le Roux, and Francis Bach. “Minimizing finite sums with the stochastic average gradient.” Mathematical Programming 162, no. 1 (2017): 83–112.

[49] Qiu, Jiayu, Xinchao Wang, Pascal Fua, and Dacheng Tao. “Matching sequences: An unsupervised approach for locality preserving sequence matching.” IEEE transactions on pattern analysis and machine intelligence (2019).

[50] Chang, Chien-Yi, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles. “D3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation.” In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3546–3555. 2019.

[51] Li, Jun, and Sinisa Todorovic. “Set-Constrained Viterbi for Set-Supervised Action Segmentation.” In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10820–10829. 2020.

[52] Shimodaira, Hiroshi, Ken-ichi Noma, Mitsuuru Nakai, and Shigeki Sagayama. “Dynamic time-alignment kernel in support vector machine.” Advances in neural information processing systems 14 (2001): 921–928.

[53] Gorelick, Lenna, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. “Actions as space-time shapes.” IEEE transactions on pattern analysis and machine intelligence 29, no. 12 (2007): 2247–2253.

[54] Hoai, Minh, and Fernando De la Torre. “Max-margin early event detectors” International Journal of Computer Vision 107, no. 2 (2014): 191–202.

[55] Pantic, Maja, Michel Valstar, Ron Rademaker, and Ludo Maat. “Web-based database for facial expression analysis.” In 2005 IEEE international conference on multimedia and Expo, pp. 5–pp. IEEE, 2005.

[56] Huang, Zongben, Ming Li, and Sinisa Todorovic. “Subspace clustering for sequentially Unaligned Data.” IEEE Access 8 (2020): 217928–217946.

[57] King, Davis E. “Dlib: A machine learning toolkit.” Journal of Machine Learning Research 10, no. Jul (2009a): 1755–1758.

[58] Warden, Pete. “Speech commands: A dataset for limited-vocabulary speech recognition.” arXiv preprint arXiv:1804.03209 (2018).

[59] McMahan, Brian, and Delip Rao. “Listening to the world improves speech command recognition.” In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1. 2018.

[60] Davis, Steven, and Paul Mermelstein. “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences.” IEEE transactions on acoustics, speech, and signal processing 28, no. 4 (1980): 355–361.

[61] LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based learning applied to document recognition.” Proceedings of the IEEE 86, no. 11 (1998): 2278–2324.

[62] Casale, Pierluigi, Oriol Pujol, and Petia Radeva. “Personalization and user verification in wearable systems using biometric walking patterns.” Personal and Ubiquitous Computing 16, no. 5 (2012): 563–580.

[63] Cai, Deng, Xiaofei He, and Jiawei Han. “Document clustering using locality preserving indexing.” IEEE Transactions on Knowledge and Data Engineering 17, no. 12 (2005): 1624–1637.

[64] L. Lovasz and M. Plummer. “Matching Theory.” North Holland, Budapest: Akademiai Kiado, 1986.