Differentiable Models for Sequential Representation Learning and Segmentation

by

DOAN PHONG TUNG

Dissertation

submitted to the Department of Informatics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

The Graduate University for Advanced Studies, SOKENDAI
September 2021

Committee

Advisor:	Dr. Atsuhiro TAKASU Professor of National Institute of Informatics/SOKENDAI
Subadvisor:	Dr. Kenro AIHARA Associate Professor of National Institute of Informatics/SOKENDAI
Examiner:	Dr. Yusheng JI Professor of National Institute of Informatics/SOKENDAI
Examiner:	Dr. Junichi YAMAGISHI Professor of National Institute of Informatics/SOKENDAI
Examiner:	Dr. Nobutaka ONO Professor of Tokyo Metropolitan University

Abstract

Sequential data is becoming more and more ubiquitous in a wide range of fields and application scenarios due to the advance of technologies in data collection and storage. The collected data sequences are often of high dimension and large volume that impede performances of the machine learning algorithms. Fortunately, the underlying process of the data could be much simpler and we can compress them in terms of both dimensionality and volume. In this dissertation, we focus on sequential representation learning and segmentation – two different but related research fields in sequential analysis for obtaining compressed and concise representation for sequential data. More precise, sequential representation learning compresses the dimensionality of the data into a smaller space regarding order relations among data samples in the sequence. Meanwhile, sequential segmentation compresses the volume of the data by partitioning data samples into several non-overlapping and homogeneous segments.

In sequential representation learning, it is crucial to know correspondence between data samples among different sequences. However, this information is often missing due to man-made reasons and collection device errors. Therefore, learning representation in sequential settings often requires jointly solving the alignment problem. A common challenge of the alignment and segmentation problems is their related optimization over discrete variables with combinatorial constraints. This inhibits development of efficient sequential learning models and scalable sequence segmentation methods. The aim of this dissertation is to propose models with differentiable objectives for sequential representation learning and segmentation of sequences that can succeed in dealing with the aforementioned challenge. There are three main advantages of the differentiability. First, it allows us to update all the variables in an unified manner during optimization. This is contrary to existing models where their parameters are updated alternatively. As a result, sub-optimal solutions are likely to be avoided. Second, stochastic gradient descent is now applicable to train or learn the model parameters. This helps to reduce both the time and memory complexities, enabling the models to handle large-scale sequential data. Finally, the differentiable models are now more extendable to handle multiple data sequences. The main contributions of the dissertation are:

Sequential representation learning. First, we introduce *Generalized sequential correlation analysis* (GSCA) – a deep model for multi-view learning from sequential data. Despite the fact that collected data sequences are often unaligned and the sample-wise correspondence

information is missing, GSCA can implicitly discover sample correspondence while learning representations. Thanks to the differentiable objective, the optimal alignment and representation in GSCA are obtained in a unified manner, avoiding sub-optimal solutions – a common issue of the existing models. Second, we combine GSCA with reconstruction losses of autoencoders to form the second model called *Generalized sequentially correlated autoencoders* (GSCAEs). GSCAEs provides a better trade-off between within-sequence and cross-sequence relations for sequential representation learning. Third, we develop an extension of GSCA termed *Generalized multiple sequences analysis* (GMSA) to handle multiple (more than two) data sequences.

Segmentation of sequences. First, we introduce *Kernel clustering with sigmoid-based regularization* (KCSR) – a segmentation model based on kernel clustering. With a novel sigmoid-based regularization, the objective of KCSR is smooth and can be effectively solved using gradient-based algorithms. Second, we develop a stochastic variant of KCSR termed *Stochastic kernel clustering with sigmoid-based regularization* (SKCSR). Time and memory complexities of SKCSR are much lower than those of the original KCSR and almost existing kernel-based models that prohibit them from handling extremely long data sequences. Third, we propose an extension of KCSR called *Multiple kernel clustering with sigmoid-based regularization* (MKCSR) for simultaneous segmentation of multiple data sequences.

Through extensive experimentation on different widely public datasets, performances of the proposed models were evaluated and compared with those of various baselines. The experimental results validate the advantages of our models over all the competitors.

Acknowledgements

I would like to express my appreciation to those who made this dissertation possible. My foremost thanks go to my supervisor, Prof. Atsuhiro Takasu, for his adequate guidance and support on my research. I am grateful for the freedom and patience he gave me to pursue a variety of topics. I would also like to thank Prof. Kenro Aihara, my second supervisor, and the committee members for their valuable assessment and helpful suggestions for enriching this thesis.

I would like to thank the Government of Japan for the scholarship they provided me to cover my expenses over the last five years pursuing a Ph.D. During the Ph.D. course, I have been working in a good research environment. I would like to thank the National Institute of Informatics and each individual of its staffs for their endless effort to create such a comfortable work environment and maintain the workspace. Many thanks to Takasu-lab's members and Vietnamese peoples in NII for not only discussions regarding the research experiences but also the time we set aside for each other with full of enjoyable moments and motivational conversations.

I would like to thank my cousin Vu Doan, who gave me guidance and support to pursue the academic research career. I also would like to express my sincere gratitude to Prof. Khoat Than – my previous mentor, who supported me a lot at very first time to build my background knowledge and sharpen my research skills. Last but not least, I want to thank my family members for years of support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Objectives

In the last decades, with the advances in data collection technology, the amount of data captured by various types of sensors (*e.g.*, camera, microphone, accelerometer, strain and weight sensor) has been growing exponentially. Not only being diverse in perspectives or views, these data also possess complex structure (*i.e.*, sequential order) and different feature sets (*i.e.*, dissimilar dimensions). This leads to an increasing interest in machine learning and data mining community for analyzing sequential data. The main problem associated with the analysis of these data is that they have high dimensionality and consist of a huge number of data samples. Fortunately, the underlying process of the data could be much simpler and It is possible to construct a much compressed and concise representation for the data.

Such compressed and concise construction can be obtained via representation learning and sequence segmentation. Representation learning [Bengio et al., 2013] aims at obtaining new representations of the data samples that reside in a much smaller dimensional space. As nowadays, data are often collected from various perspectives, each of which presents a view of the same data, multi-view representation learning [Sun, 2013, Zhao et al., 2017, Li et al., 2018] is introduced. It aims to exploit consistency and complementary information between these views to learn more comprehensive representations for the data. Recently, the definition of

multi-view learning has been extended to accommodate sequential data, *i.e.*, each view of the data is in the form of a sequence. To integrate information from different views, it is crucial for multi-view sequential learning methods to know sample-wise correspondence in advance. More precisely, given a sample in a particular sequence it is important to know its corresponding samples in the other sequences. However, in practice, this information is often missing as the collected data sequences are often different in length and misaligned. Therefore, multi-view sequential learning has to take correspondence problem into account. Segmentation [Bellman, 1961, Aminikhanghahi and Cook, 2017, Truong et al., 2020] takes a different approach to obtain compressed and concise representation from data sequences. It partitions data sequences into non-overlapping segments, whose total homogeneity is maximized. By doing so, the original data sequences can be summarized by a smaller number of the data representatives [Terzi and Tsaparas, 2006].

A common challenge of both sequential correspondence (*a.k.a.* alignment) and segmentation is their related optimization over discrete variables with combinatorial constraints. To obtain the optimal solutions for alignment and segmentation problems dynamic programming (DP) technique [Rabiner, 1993, Bellman, 1961] is often applied. However, this algorithm has high memory requirement ($O(n^2)$) and expensive computational complexity ($O(n^2)$, where n is the number of data samples in the given sequence). Furthermore, when jointly solving the correspondence problem along with representation learning, it is difficult to obtain good solutions. The reason is that the alignment solving and representation learning, especially for those using deep neural networks (DNNs), own different optimization schemes. Although existing models tries to mitigate this issues by iteratively updating the alignment and representation in an alternative manner. This procedure is prone to sub-optimal solutions. Several approximations for segmentation also encounter a similar problem. These methods identify boundaries between segments successively to reduce the computational time. However, because of not considering all the segments globally, there is no theoretical guarantee for the optimality of their solutions.

Objectives of this dissertation is to propose new models with differentiable objectives for multi-view sequential representation learning and segmentation of data sequence that can succeed in dealing with the aforementioned challenges. Advantages of the differentiability are as

follows: First, it allows us to simultaneously obtain all the optimal variables (those that encode the alignment and representations in multi-view sequential representation learning models or segment boundaries in sequence segmentation) instead of updating them in an alternative manner. As a result, sub-optimal solutions are likely to be avoided. Second, since the objective is differentiable we can employ stochastic gradient descent (SGD) to train or learn the model parameters. SGD estimates the gradient from only a small number of randomly sampled data points, which are often called as mini-batch. Therefore, it can reduces both computational time and memory requirement for the models. Finally, the differentiable models are easy to be extended. As will be shown in the subsequent chapters, with slight modifications, we can derive extensions of the differentiable models that can handle multiple data sequences simultaneously.

1.2 Contributions

Contributions of this dissertation are summarized as follows

1. Novel differentiable models for multi-view sequential representation learning

   (a) First, we introduce *Generalized sequential correlation analysis* (GSCA) – a deep model for representation learning from multi-view sequential data. The model can implicitly discover sample correspondence while learning representations. Furthermore, thanks to the differentiability of objective, the optimal alignment and representation in GSCA are obtained in a unified manner, avoiding sub-optimal solutions.

   (b) Second, we propose a variant of GSCA called *Generalized sequentially correlated autoencoders* (GSCAE). This is a result of the combination between the differentiable objective function in GSCA and reconstruction losses of autoencoders. GSCAE provides us a better trade-off between within-view and cross-view information for multi-view sequential representation learning.

   (c) Third, we derive an extension of GSCA called *Generalized multiple sequences analysis* (GMSA) for handling multiple (more than two) data sequences. All the proposed

models are trained using stochastic gradient descent. Thus, they can handle sequential datasets that are not only large in volumes but also long in the average length.

2. Novel differentiable models for segmenation of data sequences

   (a) Fourth, we introduce *Kernel clustering with sigmoid regularization* (KCSR) – a segmentation model based on kernel clustering. With a novel sigmoid-based regularization, objective of KCSR is smooth and differentiable almost every where w.r.t unconstrained and continuous variables. Therefore, all the optimal parameters of KCSR can be simultaneously and effectively obtained using gradient-based algorithms.

   (b) Fifth, we further derive stochastic gradient-based algorithm to optimize the objective function of KCSR. This variant is named *Stochastic KCSR* (SKCSR). SKCSR has much lower time and memory complexities than the original KCSR does. Especially, there is no need for SKCSR to store the whole kernel matrix, which is one of the most serious drawback of existing kernel-based models that inhibits them from handling exstremly long data sequences.

   (c) Sixth, we slightly modify the sigmoid-based regularization to form an extension of KCSR called *Multiple KCSR* (MKCSR). The new model that inherits many good properties from KCSR and SKCSR can simultaneously perform segmentation on multiple sequences effectively.

3. Finally, we conduct extensive experiments on various widely public datasets to evaluate performances of the proposed models. The experimental results validate advantages of the proposed methods over the corresponding competent baselines.

1.3 Organization

The rest of this dissertation is organized as follows. Chapter 2 provides some background knowledge on gradient-based optimization, optimal alignment and segmentation algorithms.

This chapter also revises previous work on multi-view representation learning from sequential data without correspondence and approximation models for kernel-based segmentation of data sequences. Chapter 3 introduces three novel deep models, including GSCA, GSCAE and GMSA, for multi-view sequential learning and provide their performances evaluation through extensive experiments. Chapter 4 proposes a novel regularization based on sigmoid functions that serves as a basis to develop KCSR, SKCSR and MKCSR for segmentation of data sequences. An application of the proposed models on vehicle detection is described in chapter 5. Chapter 6 discusses future works and concludes the thesis.

Chapter 2

Background and Related Work

This chapter consists of two main parts. The first part provides background on gradient-based optimization, optimal alignment and segmentation algorithms. These knowledge is helpful for understanding the rest of this dissertation. The second part revises the previous works on multi-view representation learning from sequential data without correspondence and approximation models for segmentation of data sequences.

2.1 Background

2.1.1 Gradient-based Optimization

The goal of learning a model [Tsypkin and Nikolic, 1971, Tsypkin, 1973] consists of finding the minimum of an *expected risk function* $J(\boldsymbol{w})$, which is decomposed as follows

$$J(\boldsymbol{w}) \coloneqq \mathbb{E}_{\boldsymbol{x}}\, Q(\boldsymbol{x}, \boldsymbol{w}) \coloneqq \int Q(\boldsymbol{x}, \boldsymbol{w})\, d\mathbb{P}(\boldsymbol{x}). \tag{2.1}$$

The minimization variable $\boldsymbol{w}$ is the model parameter, which must be adapted as a response to observing random variable data $\boldsymbol{x}$. The *loss function* $Q(\boldsymbol{x}, \boldsymbol{w})$ measures the performance of the learning model with parameter $\boldsymbol{w}$ under a particular observation of $\boldsymbol{x}$.

In practice, the probability distribution of the data is often unknown and only a set of data observations $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ is given. [Vapnik, 1982] show that minimizing the *empirical risk*, which is defined as

$$\widetilde{J}(\boldsymbol{w}) \coloneqq \frac{1}{n}\sum_{i=1}^{n} Q(\boldsymbol{x}_i, \boldsymbol{w}), \tag{2.2}$$

can provide a good estimate of the minimum of the expected risk $J(\boldsymbol{w})$. If $\widetilde{J}(\boldsymbol{w})$ is smooth and differentiable almost everywhere *w.r.t* the parameter $\boldsymbol{w}$, its minimum can be achieved using gradient decent (GD) algorithm. At each iteration, the parameter $\boldsymbol{w}$ is updated using the following formula

$$\boldsymbol{w}_{(t+1)} = \boldsymbol{w}_{(t)} - \eta_{(t)}\frac{\partial \widetilde{J}(\boldsymbol{w}_{(t)})}{\partial \boldsymbol{w}} = \boldsymbol{w}_{(t)} - \eta_{(t)}\frac{1}{n}\sum_{i=1}^{n}\frac{\partial Q(\boldsymbol{x}_i, \boldsymbol{w}_{(t)})}{\partial \boldsymbol{w}}, \tag{2.3}$$

where the learning rate $\eta_{(t)}$ has a positive value, which can be fixed over all the iteration t or properly selected using strategies, such as binary search and backtracking line search [Armijo, 1966]. This algorithm is guaranteed to converge linearly toward a local minimum of $\widetilde{J}(\boldsymbol{w})$ [Polyak, 1987].

In case n is large, computation of the gradient $\frac{\partial \widetilde{J}(\boldsymbol{w}_{(t)})}{\partial \boldsymbol{w}}$ becomes both time and memory demanding. Mini-batch stochastic gradient descent (SGD) algorithm [Bottou, 1998] can alleviate this difficulty by selecting a random subset $I_{(t)} \subseteq \{1, \ldots, n\}$ of size $|I_{(t)}| = b \ll n$ at each iteration t to update $\boldsymbol{w}$ using the following formula.

$$\boldsymbol{w}_{(t+1)} = \boldsymbol{w}_{(t)} - \eta_{(t)}\frac{1}{b}\sum_{i\in I_{(t)}}\frac{\partial Q(\boldsymbol{x}_i, \boldsymbol{w}_{(t)})}{\partial \boldsymbol{w}}. \tag{2.4}$$

Here, the full batch gradient are approximating by an unbiased estimate

$$\mathbb{E}_{\boldsymbol{x}}\left[\frac{1}{b}\sum_{i\in I_{(t)}}\frac{\partial Q(\boldsymbol{x}_i, \boldsymbol{w})}{\partial \boldsymbol{w}}\right] = \frac{\partial \widetilde{J}(\boldsymbol{w})}{\partial \boldsymbol{w}}. \tag{2.5}$$

It is clear that SGD has much lower computational time and memory requirement for each iteration than those of GD. Furthermore, SGD is robust to local optimal [Bottou, 1998] and give more frequent updates to the model parameters. Therefore, in practice, it is witnessed

that SGD converges much faster and find better solutions in comparison with the original GD [LeCun et al., 1989, LeCun et al., 1998, Hoffman et al., 2010].

2.1.2 Optimal Sequence Alignment

Dynamic Time Warping (DTW) [Rabiner, 1993] is a popular algorithm for optimally solving the alignment problem. Given two sequences $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_m] \in \mathbb{R}^{d \times m}$, DTW constructs a distance matrix $\boldsymbol{D}(\boldsymbol{X}, \boldsymbol{Y}) \in \mathbb{R}^{n \times m}$ such that the element at position (i, j), denoted by $d_{i,j}$, is the squared distance, i.e., $d_{i,j} = ||\boldsymbol{x}_i - \boldsymbol{y}_j||_2^2$. DTW defines a warping path $\boldsymbol{\pi}$ as a sequence of the indexes of the distance matrix:

$$\boldsymbol{\pi} = \langle (i_1, j_1), \ldots, (i_p, j_p) \rangle, \tag{2.6}$$

which satisfies three conditions: i) *Boundary*: $(i_1, j_1) = (1, 1)$ and $(i_p, j_p) = (n, m)$; ii) *Continuity*: $(i_{r+1} - i_r, j_{r+1} - j_r) \in \{(0, 1), (1, 0), (1, 1)\}$ where $1 \leq r \leq p - 1$; and iii) *Monotonicity*: if $1 \leq r \leq t \leq p$ then $i_r \leq i_t$ and $j_r \leq j_t$. The optimal warping path $\boldsymbol{\pi}^* = \langle (i_1^*, j_1^*), \ldots, (i_p^*, j_p^*) \rangle$, which has the smallest sum of elements in the distance matrix along the path, will serve as the alignment solution. More precise, the sample $\boldsymbol{x}_{i_r^*}$ is matched with sample $\boldsymbol{x}_{i_r^*}$ for $1 \leq r \leq p$.

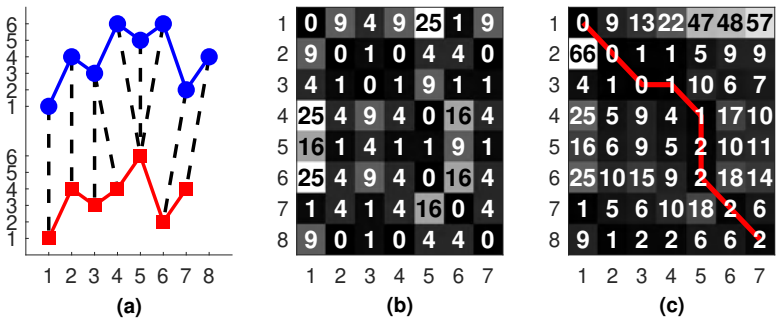


Figure 2.1: An example of DTW: (a) two example sequences, (b) the distance matrix, and (c) the cumulative sum matrix. The red line depicts the optimal warping path, which encodes the sample correspondences between the two sequences.

To achieve $\boldsymbol{\pi}^*$, the algorithm constructs an cumulative sum matrix $\boldsymbol{S}(\boldsymbol{X}, \boldsymbol{Y})$, using the

following recursive formulas:

$$s_{1,1} = d_{1,1} \tag{2.7}$$

$$s_{i,j} = d_{i,j} + \min(s_{i-1,j}, s_{i,j-1}, s_{i-1,j-1}), \tag{2.8}$$

By backtracking from the last element $s_{n,m}$ to the start element $s_{1,1}$, the optimal warping path is obtained. We also have $s_{n,m} = d_{i_1^*,j_1^*} + \cdots + d_{i_p^*,j_p^*}$ and it is defined as DTW distance between the two sequences $\text{DTW}(\boldsymbol{X}, \boldsymbol{Y}) \coloneqq s_{n,m}$. A toy example of DTW is shown in Figure 2.1.

2.1.3 Optimal Sequence Segmentation

The goal of the segmentation task is to partition a data sequence into several non-overlapping and homogeneous segments of variable durations. Let $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ 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

$$G_{j,i} = \begin{cases} 1 & \boldsymbol{x}_i \in \text{segment } j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

$\boldsymbol{G}$ must satisfy two constraints, including i) *Boundary:* $G_{1,1} = 1$ and $G_{k,n} = 1$ and ii) *Monotonicity:* for any given $G_{j,i} = 1$ then for the next column $G_{j,i+1} = 1$ or $G_{j+1,i+1} = 1$. An example of the indicator matrix is given in Figure 2.2.

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 $\kappa(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. 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 [Truong et al., 2020]. Thus, the optimization problem is often formulated as

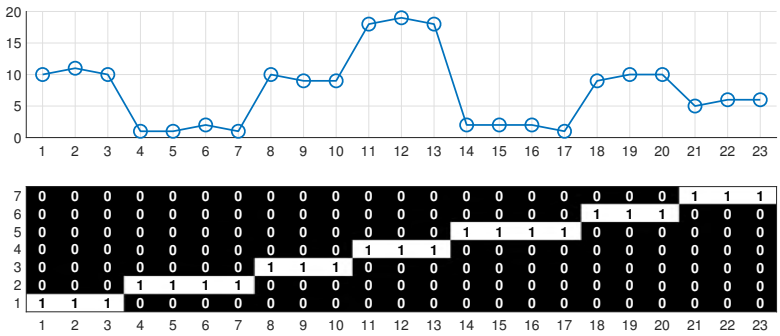


Figure 2.2: An example of sequence segmentation: (top) an example sequence of length 23 and (bottom) the corresponding indicator matrix with number of segments $k = 7$.

follows

$$\underset{\boldsymbol{G} \in \mathcal{G}}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{i=1}^{n} G_{j,i} \, ||\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_j||_2^2, \tag{2.10}$$

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.

2.2 Related Work

2.2.1 Multi-view Sequential Representation Learning

Multi-view sequential representation learning extends the definition of multi-view representation learning to accommodate sequential data, i.e. data of all the views comes in the form of sequences. There are two main approaches to develop a multi-view sequential learning model. The first approach expands single-view models for sequential data to multi-view settings. Several representatives of this approach are *Multi-view long short-term memory* (MLSTM) [Rajagopalan et al., 2016] and *Memory fusion network* (MFN) [Zadeh et al., 2018]. These models extend LSTM [Hochreiter and Schmidhuber, 1997] – a single-view sequential model to handle multi-view data. A crucial drawback of these models is that they assume sequences of different views are equal in length. However, this fragile assumption is likely to be violated in practice. For example, when collecting data sequences, sample deletion and/or insertion often

occurs due to temporal failures of devices and man-made reasons. In addition, asynchronization of data collection devices, e.g. sensors have dissimilar sampling frequencies, also induces different numbers of collected data samples for the views. Therefore, this approach is not robust for real-world data.

The second approach combines existing multi-view models for static data with DTW to adapt to sequential settings. For examples, [Zhou and Torre, 2009, Panagakis et al., 2013, Panagakis et al., 2015, Panagakis et al., 2015, Zafeiriou et al., 2016, Jia et al., 2016] directly combines *Canonical correlation analysis* (CCA) [Hotelling, 1936] with DTW. Recall that CCA is a representative of unsupervised multi-view learning methods. It tries to find a latent subspace in which projections of the views are maximally correlated. When combining with DTW, it is now able to find a subspace such that projections of the two sequences are aligned and the learned representations of the two views are maximally correlated. Recently, [Trigeorgis et al., 2018] has proposed to combine deep CCA [Andrew et al., 2013] – a variant of CCA with DTW. Based on deep neural networks, more complex and nonlinear embeddings can be obtained. For more details about advantages of deep methods over the shallow ones, we refer the readers to a recent overview paper [Rasti et al., 2020]. However, this direct combination has several serious drawbacks. Since DTW problem is discrete and its objective is not differentiable, the alignment and representations are not optimized in a unified manner. Specifically, the new alignment is updated while fixing the representations and vice versa. Without a good initialization this update scheme is prone to be sub-optimal solutions. In addition, when DNNs are used to map the two views into the new subspace their expensive training procedures need to be performed multiple times. This makes the approach inefficient and unsuitable for extending to larger deep models.

Another group of related models for multi-view learning without correspondence is based on manifold alignment. They project data from two different but correlated manifolds to a subspace, simultaneously preserving the local structures and ensuring their closeness. A subgroup of this technique includes semi-supervised methods [Wang and Mahadevan, 2008, Wang and Mahadevan, 2013, Hong et al., 2019, Abeo et al., 2019]. They utilize several sample-wise correspondences known in advance between the manifolds while learning a new subspace.

In contrast, the second subgroup, which we will focus on in this dissertation, contains unsupervised manifold alignment methods that do not require predetermining correspondences. Since there is no prior information on the sample-wise pairing, [Wang and Mahadevan, 2009] creates connection between the two views by comparing their local geometry, which is characterized by the k-Nearest Neighbor (k-NNs). [Li et al., 2020] has recently proposed a variant of this method, where the local geometry information is measured in the fuzzy granule space instead of the original space. [Tuia et al., 2014] builds a k-NNs graph for each view and extracts a series of graph-based descriptors for each data sample. The cross-view similarity and dissimilarity matrices are then computed in the descriptors space. [Cui et al., 2014] takes a different approach to the correspondence problem. Specifically, they encode the cross-view sample-wise matching into a binary matrix, which is jointly optimized with the projection matrices of the two views. [Fan et al., 2016] further extends the correspondence matrix with an extra row and column. Their aim is to better handle outliers that has no corresponding sample from the other set. Nevertheless, these method can not take advantage of sequential order in the data to discover more accurate correspondence. In addition, they are also limited to shallow models and sensitive to noise, which corrupts the adjacency and geometric information of the data.

[Gong and Medioni, 2011, Vu et al., 2012] proposed hybrid models that combine unsupervised manifold alignment with DTW. Thus, they also inherit drawbacks from the two above approaches. We note that, recently, [Le et al., 2018] has been approaching to the problem of misalignment in multi-view sequential learning using memory-based neural networks. In stead of recovering the sample correspondence between the views, this approach stores view-specific information in a memory and makes it accessible to neural network of the other view. Although having promising results in practice, we exclude this approach because it is a supervised model, which is out of scope of this dissertation.

In this dissertation, we also consider a more challenging case where the input data comprise more than two sequences. To handle this problem, existing methods such as [Zhou and De la Torre, 2015, Wang et al., 2017] combine multiset CCA [Hasan, 2009] and an approximation of DTW where the warping path is approximated by a linear combination of monotonic basic functions. Nevertheless, these methods are less favorable to data with a complex latent structure because they

can only learn a simple linear projection for each view. In addition, how to select a proper set of monotonic basics for a particular dataset remains unclear. Another closely related work is deep discriminant analysis with time warping [Trigeorgis et al., 2018]. This method simultaneously projects and aligns the input data sequences to a given label sequence. In contrast, in our case, supervised information is unavailable. Therefore, model proposed in this work solves a more complex problem.

2.2.2 Segmentation of Data Sequences

Boundaries between the segments are sometimes called as change points to indicate that at those positions some characteristics of the sequence significantly change. Our sequence segmentation problem defined in (2.10) is equivalent to offline change point detection (CPD) [Truong et al., 2020]. It aims to detect change points when all the data samples are observed. This is to distinguish from online CPD, where the changes are detected as soon as they occur in real-time setting. In this subsection, we will review related works in the literature of offline kernel CPD and use this term as an alternative for kernel segmentation. In addition, we note that our models are related to clustering. Therefore, we also review temporal clustering that also employs clustering for segmentation of data sequences.

Offline kernel change point detection. According to [Truong et al., 2020], almost all offline kernel CPD methods attempt to optimize the objective function as defined in (2.10). 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 [Harchaoui et al., 2009a, Gretton et al., 2012, Li et al., 2015b, Li et al., 2019] 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. [Harchaoui and Cappé, 2007, Arlot et al., 2019] 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 [Celisse et al., 2018] in the feature space), it is impractical for handling long data sequences. To reduce the time complexity, [Truong et al., 2019] proposed a greedy algorithm that sequentially detects change points one at an iteration. [Celisse et al., 2018] 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 [Nocedal and Wright, 2006]. 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) [Hoai and De la Torre, 2012] and Aligned clustering analysis (ACA) [Zhou et al., 2012] divide data sequences into a set of non-overlapping short segments. These subsequences are then partitioned into k classes using unsupervised support vector machine [Hoai and De la Torre, 2012] or kernel k-means clustering [Zhou et al., 2012]. 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) $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n]$ that characterizes the underlying subspaces structures and sequential (a.k.a. temporal) information of the original data. Second, the normalized cut algorithm (Ncut) [Shi and Malik, 2000] is then utilized for segmentation of $\boldsymbol{Z}$. To preserve the sequential information in the new representation, [Tierney et al., 2014, Wu et al., 2015] 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 $\boldsymbol{z}_i$ and $\boldsymbol{z}_{i+1}$ of the two consecutive samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$, respectively, are forced to be similar. [Hu et al., 2020] further integrated a weight matrix into the linear regularization to avoid equally constraining on every

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

Table 2.1: Time and space complexities of different segmentation methods

pair of consecutive samples. Nevertheless, since the regularization is linear, it is ineffective for handling complex data structure. To leverage this issue, [Li et al., 2015a, Liu et al., 2017a] proposed manifold-based regularization that preserves the sequential information for the local neighborhood data samples. This type of regularization is more preferable [Zheng et al., 2021] as it often outperforms the linear one in most tests [Clopton et al., 2017]. 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 [Celisse et al., 2018] 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 a large dataset will be given in the next section.

Table 2.1 shows time and space complexities of several existing segmentation methods. In this table, n_{max} denotes maximum length of divided subsequences in ACA. The dimension of

the new representation $\boldsymbol{Z}$ in OSC [Wu et al., 2015] and TSC [Li et al., 2015a] is expressed by d and t denotes the total iterations needed for the algorithms converge.

Chapter 3

Differentiable Models for Multi-view Sequential Learning

Multi-view learning is concerned with machine learning problems, where data are represented by distinct feature sets or views. Recently, this definition has been extended to accommodate sequential data, i.e., each view of the data is in the form of a sequence. Multi-view sequential data poses major challenges for representation learning, including i) *absence of sample correspondence information between the views*, ii) *complex relationships among samples within each view*, and iii) *high complexity for handling multiple sequences*. In this chapter, we first introduce a generalized deep learning model that can simultaneously discover sample correspondence and capture the cross-view relationships among the data sequences. The model parameters can be obtained in an unified manner using a gradient descent-based algorithm, thanks to the differentiability of the objective. The complexity for computing the gradient is at most quadratic with regard to sequence lengths in terms of both computational time and space. Based on this model, we propose a second model by integrating the objective with reconstruction losses of autoencoders. This allows the second model to provide a better trade-off between view-specific and cross-view relationships in the data. Finally, to handle multiple (more than two) data sequences, we develop a third model along with a convergence-guaranteed optimization algorithm. Extensive experiments on public datasets demonstrate the superior performances of our

models over competing methods.

3.1 Introduction

In many real-world applications, data are often collected from various perspectives, each of which presents a view of the same data and has its own representation space and relation characteristics. Multi-view learning methods aim to exploit consistency and complementary information between these views to learn new representations for the data. These methods have better generalization ability. Recently, the definition of multi-view learning has been extended to accommodate sequential data, i.e., each view of the data is in the form of a sequence. For instance, human actions can be presented by several video sequences with different features, such as binary, Euclidean distance transform, and Poisson equation solutions [Gorelick et al., 2007] (see Figure)

Multi-view sequential learning has posed major challenges that are difficult for conventional methods to accommodate. First, most multi-view learning methods essentially rely on an assumption that all views of the data are equal in size and sample-wise matching. Here, we take canonical correlation analysis (CCA) [Hotelling, 1936] and its variants [Hardoon et al., 2004, Andrew et al., 2013, Wang et al., 2015b] as representatives. These methods project data samples from two different views into a shared subspace and then minimize the squared difference between the projections subject to whitening constraints. Thus, the two-view training data must have the following form: $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^{d_x \times n}$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_n] \in \mathbb{R}^{d_y \times n}$, where $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is a matching pair $(1 \leq i \leq n)$. However, these requirements are likely to be violated in sequential settings. Sample deletion and/or insertion often occurs when collecting data sequences because of the temporal failures of devices and other man-made reasons. In addition, the asynchronization of data collection devices, e.g., sensors have dissimilar sampling frequencies, also induces misalignment among the collected sequences. A widely used alignment algorithm, dynamic time warping (DTW) [Rabiner, 1993], can be used to match samples in correspondence as a preprocessing step before performing conventional multi-view learning

methods. Unfortunately, DTW fails when the dimensions of the two sequences vary ($d_x \neq d_y$). Second, in addition to the ambiguous cross-view relationships mentioned, multi-view sequential data also involve complex view-specific relationships that span the length of the sequences. CCA-based methods capture these relationships through linear [Hotelling, 1936] and nonlinear [Hardoon et al., 2004, Andrew et al., 2013, Wang et al., 2015b] projection functions. However, they ignore the sequential order that naturally exists among samples within each view. Finally, in practice, the input data often comprise more than two sequences. Handling multiple sequential views is a difficult task that certainly involves high resource requirements. In addition, the discriminative properties of the learned representation might be degenerated because of the absence of label information and the presence of irrelevant information from multiple views.

In this chapter, we first propose generalized sequential correlation analysis (GSCA)—a novel deep neural network (DNN)-based model—to tackle the aforementioned challenges. Our model parameterizes the projection functions that map data sequences into the shared subspace by DNNs. Various types of DNNs can be selected regarding the relationships among samples within each view. In this work, we use feed-forward neural networks and recurrent neural networks (RNNs) for implementation. In the shared subspace, our model minimizes the generalized smooth DTW distance between projections of the two views subject to soft whitening constraints. This allows GSCA to discover the sample correspondences and capture the relationships between the views simultaneously. Because the generalized smooth DTW is a differentiable approximation of the original DTW, parameters of our model can be optimized in a unified manner using gradient descent-based algorithms. Computing the gradient generally takes a quadratic time and requires a quadratic memory space considering the sequence lengths. We can further increase the computation speed and reduce the space requirement by selecting squared ℓ_2 norm for regularization, which induces sparsity in the gradient of the generalized smooth DTW. Second, to provide a better balance between view-specific and cross-view relationships, we combine our objective function with the reconstruction losses of autoencoders [Ngiam et al., 2011]. This forms generalized sequentially correlated autoencoders (GSCAEs), which are a new variant of the proposed model. Finally, we further develop the third model called generalized multiple sequences analysis (GMSA) to handle multiple data se-

quences. Slightly differing from the two first proposed models, GMSA uses DNNs to map input data sequences directly into the label space. Thereby, we expect that the learned representation can have cluster interpretability and better discriminability. Because no supervised information is given, we introduce a consensus label sequence that is then aligned with projections of all the input sequences. An efficient algorithm with a convergence guarantee is also provided to optimize both the consensus and the DNNs' parameters.

The remainder of this chapter is organized as follows: Section 3.2 briefly presents some background for the models proposed in this chapter. The GSCA model and its autoencoder-based variant are introduced in Sections 3.3 and 3.4, respectively. The third model for handling multiple data sequences along with a convergence-guaranteed optimization algorithm is described in Section 3.5. After reporting the experimental results in Section 3.6, we summarize the chapter in Section 3.7.

3.2 Preliminary

3.2.1 Notations

Throughout this chapter, scalars, vectors, and matrices are denoted by lower-case, bold lower-case, and bold uppercase letters, respectively. An element at position (i, j) of a matrix $\boldsymbol{A}$ is denoted by $a_{i,j}$ or $[\boldsymbol{A}]_{i,j}$. We denote the Frobenius inner product between $\boldsymbol{A}$ and $\boldsymbol{B}$ as $\langle \boldsymbol{A}, \boldsymbol{B} \rangle \coloneqq \sum_{i,j} a_{i,j} b_{i,j}$. $\mathbf{0}_d$ is a vector of dimension d whose all elements are zeros. The expression $\boldsymbol{x} \in \mathbb{R}_+^d$ indicates that vector $\boldsymbol{x}$ has d elements, each of which is greater than or equal to zero. The norm ℓ_p of a vector $\boldsymbol{x}$, where $p \in \{1, 2\}$ in this chapter, is $||\boldsymbol{x}||_p = (|x_1|^p + \ldots |x_d|^p)^{\frac{1}{p}}$.

3.2.2 Generalized Smooth DTW

The optimal warping path can be discovered by minimizing DTW; however, original DTW is not differentiable because of the nonsmoothness of min operator in equation (2.8), which makes

it difficult to minimize using gradient-based methods. To alleviate this issue, [Nesterov, 2005, Mensch and Blondel, 2018] studied a smooth min operator that serves as an essential basis to develop the differentiable approximations of DTW.

Let $\boldsymbol{\eta} = [\eta_1, ..., \eta_k]^\top \in \mathbb{R}^k$, the smooth min operator is defined as follows:

$$\min_\Omega(\boldsymbol{\eta}) \coloneqq \min_{\boldsymbol{\gamma} \in \Delta^k} \langle \boldsymbol{\gamma}, \boldsymbol{\eta} \rangle + \frac{1}{\beta}\Omega(\boldsymbol{\gamma}), \tag{3.1}$$

where $\Delta^k \coloneqq \{\boldsymbol{\gamma} \in \mathbb{R}_+^k : ||\boldsymbol{\gamma}||_1 = 1\}$ is a $(k-1)$ unit simplex, $\langle ., . \rangle$ denotes an inner product, Ω is a strictly convex function on Δ^k, and β is a nonnegative regularization parameter. Because (3.1) is strictly convex, its minimum is unique and equal to the gradient (based on Danskin's theorem [Danskin, 1966]):

$$\nabla \min_\Omega(\boldsymbol{\eta}) = \operatorname*{argmin}_{\boldsymbol{\gamma} \in \Delta^k} \langle \boldsymbol{\gamma}, \boldsymbol{\eta} \rangle + \frac{1}{\beta}\Omega(\boldsymbol{\gamma}). \tag{3.2}$$

The equation shows that the smooth min operator also depends on the selection of the regularization function $\Omega(\boldsymbol{\gamma})$. Shannon entropy ($\sum_{i=1}^k \gamma_i \ln \gamma_i$) or squared ℓ_2 norm ($\frac{1}{2}\sum_{i=1}^k \gamma_i^2$) are often chosen. While the former induces closed-form solutions for both smooth min and its gradient, the latter forces the gradient to be sparse. More details are given in Appendix A.

As the definition of the smooth min operator is already given, we can arrive at the following recursive formulation:

$$\begin{aligned} s'_{1,1} &= d_{1,1} \\ s'_{i,j} &= d_{i,j} + \min_\Omega(s'_{i-1,j}, s'_{i,j-1}, s'_{i-1,j-1}), \end{aligned} \tag{3.3}$$

where the generalized smooth approximation of DTW is defined by $\mathrm{DTW}_\Omega(\boldsymbol{X}, \boldsymbol{Y}) \coloneqq s'_{n,m}$. Note that we can have different versions of DTW_Ω, e.g., $\mathrm{DTW}_{\Omega=\text{entropy}}$ or $\mathrm{DTW}_{\Omega=\text{squared}\ell_2}$, depending on selection of the regularization $\Omega(\boldsymbol{\gamma})$. The generalized smooth DTW distance is different from the original DTW because it is differentiable. Furthermore, by minimizing DTW_Ω, the optimal warping path is discovered implicitly instead of specified directly, as in the

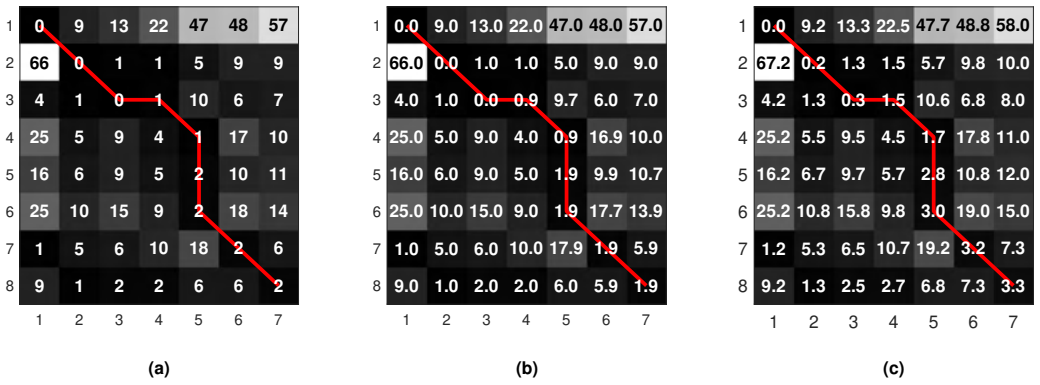


Figure 3.1: An example of Generalized smooth DTW: (a) the cumulative sum matrix of the original DTW (which is the same as in Figure 2.1 (c)), (b) the cumulative sum matrix of $\text{DTW}_{\Omega=\text{entropy}}$, and (c) the cumulative sum matrix of $\text{DTW}_{\Omega=\text{squared } \ell_2}$. The red line depicts the optimal warping path, which encodes the sample correspondences between the two sequences.

original DTW. An example of Generalized Smooth DTW is illustrated in Figure 3.1.

3.3 Generalized Sequential Correlation Analysis

In this section, we propose a method, namely *Generalized sequential correlation analysis* (GSCA), for multi-view representation learning from sequential data. We first present the model and its objective function. We then describe the optimization methods and the relationship between the proposed method and *Deep canonical correlation analysis* DCCA [Andrew et al., 2013].

Given two data sequences $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^{d_x \times n}$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_m] \in \mathbb{R}^{d_y \times m}$ from different representation spaces ($d_x \neq d_y$), our method maps them into a shared subspace: $\boldsymbol{Z}^x = [\boldsymbol{z}_1^x, ..., \boldsymbol{z}_n^x] = f_x(\boldsymbol{X}, \boldsymbol{\theta}_x) \in \mathbb{R}^{d \times n}$ and $\boldsymbol{Z}^y = [\boldsymbol{z}_1^y, ..., \boldsymbol{z}_m^y] = f_y(\boldsymbol{Y}, \boldsymbol{\theta}_y) \in \mathbb{R}^{d \times m}$, where $f_x(\cdot, \cdot)$ and $f_y(\cdot, \cdot)$ are projection functions and $\boldsymbol{\theta_x}$ and $\boldsymbol{\theta_y}$ denotes their parameters. The projection functions are parameterized by deep feed-forward neural networks or RNNs. If the former is selected, each sample $\boldsymbol{x}_i$ of the sequence $\boldsymbol{X}$ is passed through several fully connected feed-forward layers to compute the output $\boldsymbol{z}_i^x$. These outputs are then assembled into columns of the matrix $\boldsymbol{Z}^x$ following the increasing order of the index i. The second view is processed in the same manner. Note that we use BN [Ioffe and Szegedy, 2015] as the final layer. Thus, the output features have zero mean and unit variance. For the latter, we stack several LSTM units to form two deep LSTM networks. Each network is also equipped with a BN layer to perform

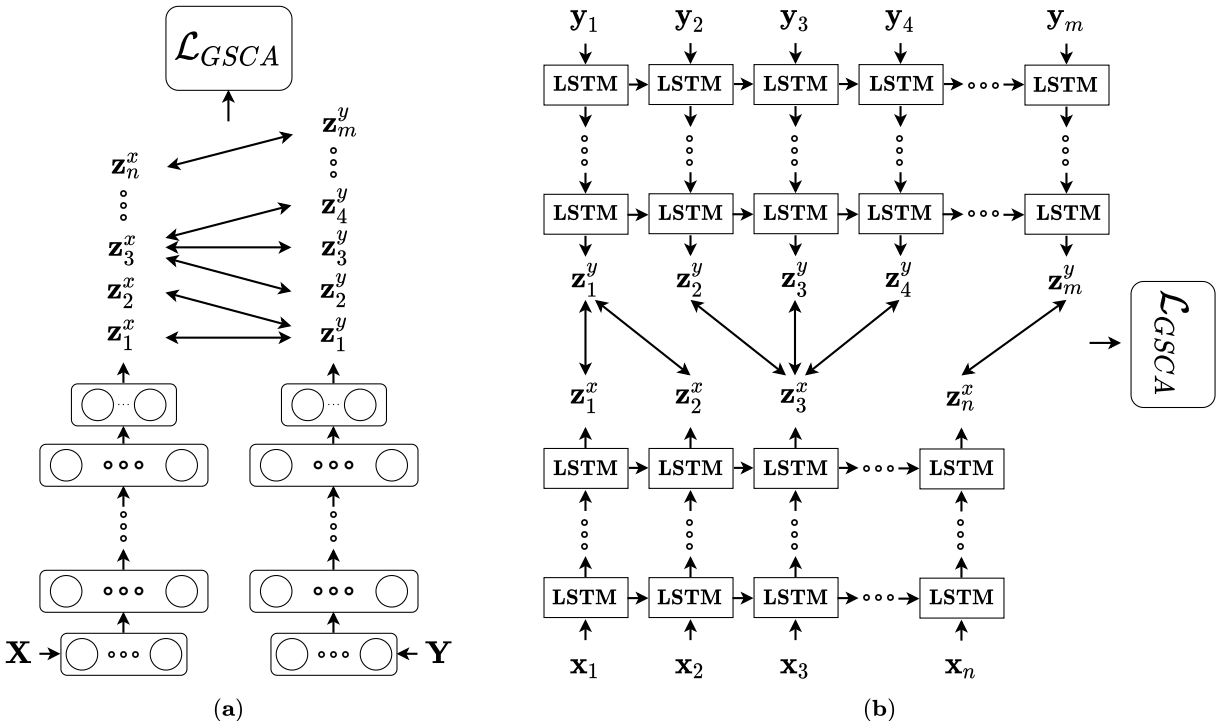


Figure 3.2: Diagrams of GSCA, where the projection functions are parameterized by (a) deep feed-forward neural networks or (b) deep RNNs (unfolded deep LSTM networks are shown). The symbol $\leftrightarrow$ denotes the sample correspondences that are discovered implicitly by minimizing the objective $\mathcal{L}_{GSCA}$. Note that each deep network includes a batch normalization (BN) layer at the output, which is not shown in the diagrams.

the normalization. The representation sequences $\boldsymbol{Z}^x$ and $\boldsymbol{Z}^y$ are then computed by feeding the input sequence $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively, to the networks. Note that the parameters $\boldsymbol{\theta_x}$ and $\boldsymbol{\theta_y}$ are now equivalent to collections of all the weights matrices of the corresponding DNNs. Figure 3.2 shows the diagrams of the proposed method.

3.3.1 Objective

Because the input sequences are unaligned, the sample-wise correspondence information between their representations $\boldsymbol{Z}^x$ and $\boldsymbol{Z}^y$ is also absent. Our model aims at minimizing the generalized smooth DTW distance between $\boldsymbol{Z}^x$ and $\boldsymbol{Z}^y$. This allows the model to discover the optimal warping path implicitly by encoding the sample correspondences as mentioned in Section 3.2. In addition, the squared distances between the corresponding representation samples from the two views are also reduced simultaneously, pulling them closer in the shared subspace.

The objective function of our model is as follows:

$$\mathcal{L}_{GSCA} = \mathrm{DTW}_{\Omega}(\boldsymbol{Z}^x, \boldsymbol{Z}^y) + \lambda_1 \mathcal{L}_x(\boldsymbol{Z}^x) + \lambda_2 \mathcal{L}_y(\boldsymbol{Z}^y), \tag{3.4}$$

where the two regularization terms are of the following form:

$$\mathcal{L}_v(\boldsymbol{Z}^v) = \sum_{i=1}^{d} \sum_{j \neq i}^{d} |c_{i,j}^v|, \tag{3.5}$$

where $v \in \{x, y\}$ and $c_{i,j}^v$ is the element at the (i, j) position of the matrix $\boldsymbol{C}^v = \boldsymbol{Z}^v \boldsymbol{Z}^{v\top}$. These regularization functions are smooth approximations of the whitening constraints in CCA-based methods. More specifically, the whitening constraints enforce the features of the representations to be pairwise uncorrelated ($\boldsymbol{C}^v = \boldsymbol{I}$). They are used to prevent trivial solutions, e.g., all the data samples are mapped into a single point in the shared subspace. In our method, because the representation sequences are normalized by BN layers, we further use the l_1-norm to encourage sparsity in the off-diagonal elements of $\boldsymbol{C}^v$. $\lambda_1 > 0$ and $\lambda_2 > 0$ are regularization parameters that control the trade-off between whitening and warping the two representation sequences.

3.3.2 Optimization

The parameters $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_y$ can be trained using the gradient-based method. To compute the gradient of $\mathcal{L}_{GSCA}$ with regard to all the parameters $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_y$, we compute its gradients with regard to the outputs $\boldsymbol{Z}^x$ and $\boldsymbol{Z}^y$ and then use backpropagation [LeCun et al., 1989] in the case of feed-forward neural networks or backpropagation through time (BTT) [Werbos, 1990] if the RNNs are used. We have

$$\frac{\partial \mathcal{L}_{GSCA}}{\partial \boldsymbol{Z}^x} = \frac{\partial \mathrm{DTW}_{\Omega}(\boldsymbol{Z}^x, \boldsymbol{Z}^y)}{\partial \boldsymbol{Z}^x} + \lambda_1 \frac{\partial \mathcal{L}_x(\boldsymbol{Z}^x)}{\partial \boldsymbol{Z}^x}. \tag{3.6}$$

The gradient of the generalized smooth DTW with regard to $\boldsymbol{Z}^x$ can be computed as

$$\frac{\partial \mathrm{DTW}_{\Omega}(\boldsymbol{Z}^x, \boldsymbol{Z}^y)}{\partial \boldsymbol{Z}^x} = \left[\frac{\partial s'_{n,m}}{\partial \boldsymbol{z}_1^x}, \ldots, \frac{\partial s'_{n,m}}{\partial \boldsymbol{z}_n^x} \right], \tag{3.7}$$

Algorithm 1 : Stochastic algorithm for GSCA

Require: Batch size ratio $\alpha \in [0, 1]$, time constant $\rho \in [0, 1]$, momentum $\mu \in [0, 1)$, and learning rate ϵ.

Ensure: Optimal DNNs parameters $\boldsymbol{\theta}^* = [\boldsymbol{\theta}_x^*, \boldsymbol{\theta}_y^*]$.

for $t = 1, \ldots, \mathrm{T}$ **do**
  random sample subsequence $\boldsymbol{Z}_{(t)}^x$ of length $n\alpha$;
  random sample subsequence $\boldsymbol{Z}_{(t)}^y$ of length $m\alpha$;
  $\boldsymbol{C}_{(t)}^x = \rho \boldsymbol{C}_{(t-1)}^x + (1-\rho)\frac{1}{\alpha}\boldsymbol{Z}_{(t)}^x \boldsymbol{Z}_{(t)}^{x\top}$;
  $\boldsymbol{C}_{(t)}^y = \rho \boldsymbol{C}_{(t-1)}^y + (1-\rho)\frac{1}{\alpha}\boldsymbol{Z}_{(t)}^y \boldsymbol{Z}_{(t)}^{y\top}$;
  compute $\frac{\partial \mathcal{L}_{GSCA}}{\partial \boldsymbol{Z}_{(t)}^x}$ and $\frac{\partial \mathcal{L}_{GSCA}}{\partial \boldsymbol{Z}_{(t)}^y}$;
  compute gradient $\nabla_{\boldsymbol{\theta}}$ using backpropagation;
  $\Delta\boldsymbol{\theta}_{(t)} = \mu\Delta\boldsymbol{\theta}_{(t-1)} - \epsilon\nabla_{\boldsymbol{\theta}}$;
  $\boldsymbol{\theta}_{(t)} = \boldsymbol{\theta}_{(t-1)} + \Delta\boldsymbol{\theta}_{(t)}$;
end for

where

$$\frac{\partial s'_{n,m}}{\partial \boldsymbol{z}_i^x} = \sum_{j=1}^{m} \frac{\partial s'_{n,m}}{\partial d_{i,j}} \frac{\partial d_{i,j}}{\partial \boldsymbol{z}_i^x} \tag{3.8}$$

$$= 2\sum_{j=1}^{m} e_{i,j}\left(\boldsymbol{z}_i^x - \boldsymbol{z}_j^y\right) \qquad \text{for } i = 1, ..., n. \tag{3.9}$$

In equation (3.9), we used the notations defined in Section 3.2, where $s'_{n,m} \coloneqq \mathrm{DTW}_\Omega(\boldsymbol{Z}^x, \boldsymbol{Z}^y)$ and $d_{i,j} \coloneqq ||\boldsymbol{z}_i^x - \boldsymbol{z}_j^y||_2^2$. The derivative $e_{i,j} = \frac{\partial s'_{n,m}}{\partial d_{i,j}}$ can be computed efficiently using a forward-backward algorithm. The details of the algorithm and its complexity are given in Appendix C.

The gradient of $\mathcal{L}_x$ with regard to $\boldsymbol{Z}^x$ can be computed as

$$\frac{\partial \mathcal{L}_x(\boldsymbol{Z}^x)}{\partial \boldsymbol{Z}^x} = \boldsymbol{H}^x \boldsymbol{Z}^x, \tag{3.10}$$

where $\boldsymbol{H}^x \in \mathbb{R}^{d\times d}$, whose elements are defined as

$$h_{i,j}^x = \begin{cases} 1 & \text{if } c_{i,j}^x > 0 \\ 0 & \text{if } i = j \text{ or } c_{i,j}^x = 0 \\ -1 & \text{if } c_{i,j}^x < 0. \end{cases} \tag{3.11}$$

The gradient $\frac{\partial \mathcal{L}_{GSCA}}{\partial \boldsymbol{Z}^y}$ can be computed in a similar manner. Our model can be trained using a full-batch algorithm (L-BFGS) [Nocedal and Wright, 2006], as in [Andrew et al., 2013]. For large datasets, however, this algorithm is both time and memory inefficient. An alternative is based on stochastic gradient descent (SGD) [Bottou, 1991, LeCun et al., 1998] where the gradient is estimated based on a much smaller number of training samples (a minibatch). The details are shown in Algorithm 1. Note that we use a stochastic estimate of the covariance matrix for each view because at each iteration, t, the algorithm can access only a small number of samples instead of the whole training set.

3.3.3 Relation to Deep CCA

Let us consider the case where the data sequences X and Y are equal in size ($m = n$), then the DTW distance between $\boldsymbol{Z}^x$ and $\boldsymbol{Z}^y$ is equivalent to their squared difference. By replacing the two regulation terms in the objective (3.4) with their associated hard whitening constraints, the optimization problem of GSCA model turns into

$$\begin{aligned} \min_{\boldsymbol{\theta}_x, \boldsymbol{\theta}_y} \quad & ||\boldsymbol{Z}^x - \boldsymbol{Z}^y||_F^2 \\ \text{s.t.} \quad & \boldsymbol{Z}^x \boldsymbol{Z}^{x\top} = \boldsymbol{Z}^y \boldsymbol{Z}^{y\top} = \boldsymbol{I}, \end{aligned} \tag{3.12}$$

which is exactly the optimization problem of DCCA. This indicates that GSCA also maximizes the correlation between the projections of the views. However, our models is more generalized than DCCA, since it can handle multi-view sequential data, which are possibly unequal in size and misaligned.

3.4 Generalized Sequentially Correlated Autoencoders

In this section, we develop GSCAEs as a variant of the proposed method. The objective of GSCAEs is formed by integrating reconstruction losses of autoencoders with the objective of GSCA. Let $\mathrm{g}_x(\boldsymbol{Z}^x, \boldsymbol{\Phi}_x)$ and $\mathrm{g}_y(\boldsymbol{Z}^y, \boldsymbol{\Phi}_y)$ denote the functions that map the representation

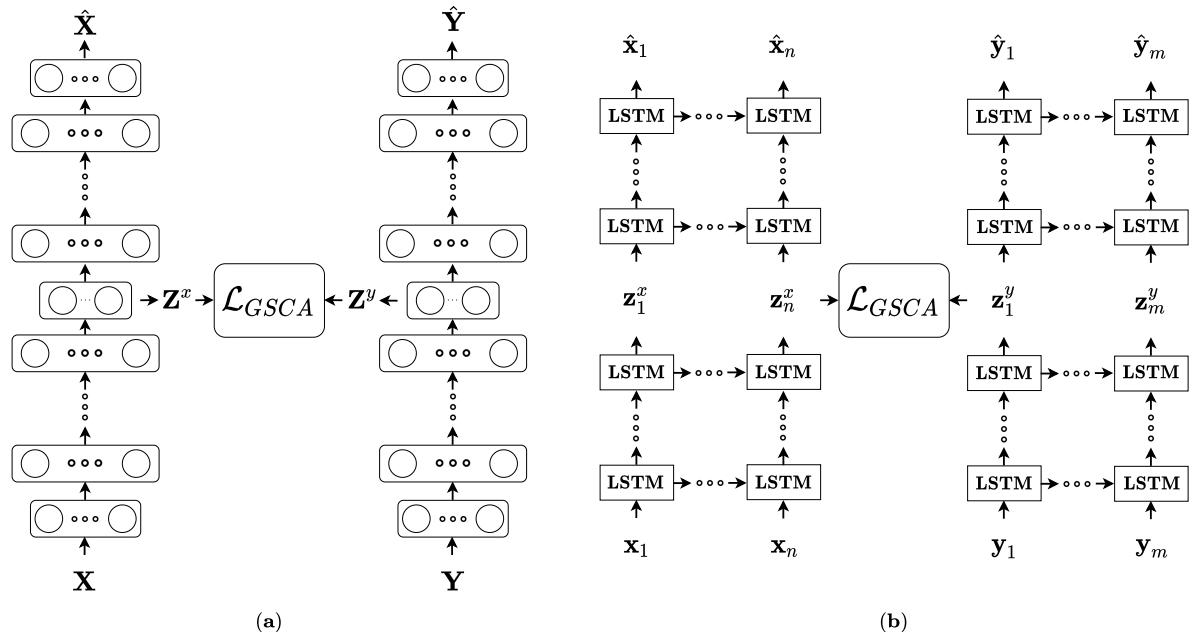


Figure 3.3: Diagrams of *generalized sequentially correlated autoencoders* (GSCAEs), where the projections functions and reconstruction functions are parameterized by (a) deep feed-forward neural networks or (b) deep RNNs. $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}_1, ..., \hat{\boldsymbol{x}}_N]$ and $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_M]$ denote the reconstructed inputs.

sequences $\boldsymbol{Z}^x$ and $\boldsymbol{Z}^y$ back to the original spaces, where $\boldsymbol{\Phi}_x$ and $\boldsymbol{\Phi}_y$ are their corresponding parameters. Then, the objective of GSCAEs is as follows

$$\mathcal{L}_{GSCAE} = \mathcal{L}_{GSCA} + \lambda \left(\frac{1}{n} ||\boldsymbol{X} - \mathrm{g}_x(\boldsymbol{Z}^x, \boldsymbol{\Phi}_x)||_2^2 + \frac{1}{m} ||\boldsymbol{Y} - \mathrm{g}_y(\boldsymbol{Z}^y, \boldsymbol{\Phi}_y)||_2^2 \right), \tag{3.13}$$

where $\lambda > 0$ is a trade-off parameter. Similar to projection functions in GSCA, $\mathrm{g}_x(\cdot, \cdot)$ and $\mathrm{g}_y(\cdot, \cdot)$ can also be parameterized by deep feed-forward neural networks or RNNs. Diagrams of GSCAEs are illustrated in Figure 3.3.

By minimizing $\mathcal{L}_{GSCA}$, the correspondences of samples between the views are discovered implicitly and their corresponding squared distances are also reduced. This amounts to maximizing mutual information, which presents the relation between the views. The view-specific relation, on the other hand, is expressed via minimizing the reconstruction errors. This is equivalent to maximizing a bound on the mutual information between the input and output of each view. Thus, GSCAEs provide us a better trade-off between information within each view and cross-view information.

The advantages of GSCAEs over GSCA come at some expenses. For a specific application with a particular dataset, we need to carefully tune the trade-off parameter λ for GSCAEs to achieve optimal performance. In addition, training GSCAEs certainly requires more computational resources than those for GSCA. Specifically, when training GSCAEs using a stochastic-based algorithm, we need to compute the gradients with regard to $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_y$, which are associated with both $\mathcal{L}_{GSCA}$ and autoencoder parts. Furthermore, we also need to compute the gradients with regard to $\boldsymbol{\Phi}_x$ and $\boldsymbol{\Phi}_y$, which are only dependent on the reconstruction losses. We note that similar to GSCA, GSCAEs have different versions depending on the selection of the regularization function $\Omega(\boldsymbol{\gamma})$ in DTW_Ω. We denote them as GSCAE-e if $\Omega(\boldsymbol{\gamma})$ is Shannon entropy and GSCAE-s when squared ℓ_2 norm is selected.

3.5 Generalized Multiple Sequences Analysis

In this section, we propose generalized multiple sequences analysis (GMSA), which is an extended variant of GSCA for learning representation using multiple data sequences. Slightly differing from the previously proposed method, GMSA directly projects all the data sequences into the label subspace to learn more interpretable and discriminative representations. The projection functions are parameterized by DNNs, as in GSCA. To accommodate sequential mismatching, we introduce a consensus label sequence that is then aligned to all the output sequences of the DNNs. An alternating optimization algorithm is finally derived to solve the objective function with regard to parameters of the DNNs and the consensus label sequence.

3.5.1 Objective

Given v data sequences $\boldsymbol{X}^{(k)} \in \mathbb{R}^{d_{x^{(k)}} \times n^{(k)}}$ for $k = 1, ..., v$, we assume that each sample of these sequences belongs to one of c disjoint classes. The cluster assignment is often denoted by a matrix $\boldsymbol{F}^{(k)} = [\boldsymbol{f}_1^{(k)}, ..., \boldsymbol{f}_{n^{(k)}}^{(k)}] \in \mathbb{R}^{c \times n^{(k)}}$, where $\boldsymbol{f}_i^{(k)}$ is the cluster indicator vector [1] of sample $\boldsymbol{x}_i^{(k)}$ in the sequence $\boldsymbol{X}^{(k)}$. As in [Ye et al., 2007, Nie et al., 2011, De la Torre, 2012], for each

[1] $\boldsymbol{f}_i^{(k)} \in \{0, 1\}^{c \times 1}$ such that $f_{j,i}^{(k)} = 1$ if $\boldsymbol{x}_i^{(k)}$ belongs to the j^{th} class and zero otherwise.

view, we define a scaled cluster indicator matrix

$$\widetilde{\boldsymbol{F}}^{(k)} = [\widetilde{\boldsymbol{f}}_1^{(k)}, ..., \widetilde{\boldsymbol{f}}_{n^{(k)}}^{(k)}] = (\boldsymbol{F}^{(k)^\top}\boldsymbol{F}^{(k)})^{-\frac{1}{2}}\boldsymbol{F}^{(k)^\top}. \tag{3.14}$$

It turns out that

$$\widetilde{\boldsymbol{F}}^{(k)} \geq \boldsymbol{0} \;\text{ and }\; \widetilde{\boldsymbol{F}}^{(k)}\widetilde{\boldsymbol{F}}^{(k)^\top} = \boldsymbol{I}. \tag{3.15}$$

In GMSA, each input data sequence is passed through a DNN to compute its output sequence $\boldsymbol{Z}^{(k)} = f_k(\boldsymbol{X}^{(k)}, \boldsymbol{\theta}^{(k)}) \in \mathbb{R}^{c\times n^{(k)}}$, where $\boldsymbol{\theta}^{(k)}$ is a collection of all parameters of the k^{th} network. The dimension of the new subspace is exactly the number of the classes, indicating that the input sequences are mapped into the same space with the labels. Because the representation sequences in the new space are possibly unequal in length and sample-wise mismatched, we introduce a consensus label sequence $\boldsymbol{Z} \in \mathbb{R}^{c\times n}$ of a prespecified length n and minimize its DTW distances with all sequences $\boldsymbol{Z}^{(k)}$. The optimization problem of GMSA is as follows:

$$\begin{aligned} \min_{\boldsymbol{Z},\boldsymbol{\theta}^{(1)},...,\boldsymbol{\theta}^{(v)}} \quad & \sum_{k=1}^{v} \mathrm{DTW}_\Omega(\boldsymbol{Z}, \boldsymbol{Z}^{(k)}) + \sum_{k=1}^{v} \lambda_k \mathcal{L}_k(\boldsymbol{Z}^{(k)}), \\ \text{subject to} \quad & \boldsymbol{Z} \geq \boldsymbol{0} \;\text{ and }\; \boldsymbol{Z}\boldsymbol{Z}^\top = \boldsymbol{I}, \end{aligned} \tag{3.16}$$

where λ_k is the weighted parameter for soft whitening regularization of the k^{th} view, which is similar to GSCA. The nonnegative and orthogonal constraints are derived from (3.15), enforcing $\boldsymbol{Z}$ to satisfy the cluster indicator conditions. By introducing the consensus label sequence, we can avoid minimizing the sum of all pairwise DTW distances between output sequences, which is computationally demanding and prone to errors. We note that *deep discriminant analysis with time warping* in [Trigeorgis et al., 2018] also utilizes the idea of mapping the input data sequences into the label space. However, the authors assumed that the supervised information was already available. In our model, the sequential labels are not given previously. Therefore, GMSA is completely unsupervised and its optimization problem is more difficult to be solved.

3.5.2 Optimization

In this section, we propose an alternating algorithm to solve the optimization problem of GMSA. More specifically, we update all the parameters of the DNNs iteratively when $\boldsymbol{Z}$ is fixed and then optimize the consensus label sequence after recomputing all output sequences of the DNNs.

Let $\mathcal{L}_{GMSA}$ denote the objective function in (3.16). When fixing the consensus label sequence, we can compute gradients of $\mathcal{L}_{GMSA}$ with regard to $\boldsymbol{Z}^{(k)}$ for $k = 1, ..., v$ as follows:

$$\frac{\partial \mathcal{L}_{GMSA}(\boldsymbol{Z}^{(k)})}{\partial \boldsymbol{Z}^{(k)}} = \frac{\partial \mathrm{DTW}_{\Omega}(\boldsymbol{Z}, \boldsymbol{Z}^{(k)})}{\partial \boldsymbol{Z}^{(k)}} + \lambda_k \frac{\partial \mathcal{L}_k(\boldsymbol{Z}^{(k)})}{\partial \boldsymbol{Z}^{(k)}}. \tag{3.17}$$

Similar to optimization for GSCA, these gradients are then backpropagated to compute the gradients of $\mathcal{L}_{GMSA}$ with regard to the parameters $\boldsymbol{\theta}^{(k)}$ for $k = 1, ..., v$.

When all parameters $\boldsymbol{\theta}^{(k)}$ are fixed, the output sequences $\boldsymbol{Z}^{(k)}$ are recomputed and the optimization problem in (3.16) reduces to

$$\begin{aligned} \min_{\boldsymbol{Z}} \quad & \sum_{k=1}^{v} \mathrm{DTW}_{\Omega}(\boldsymbol{Z}, \boldsymbol{Z}^{(k)}) \\ \text{subject to} \quad & \boldsymbol{Z} \geq \boldsymbol{0} \ \text{ and } \ \boldsymbol{Z}\boldsymbol{Z}^{\top} = \boldsymbol{I}. \end{aligned} \tag{3.18}$$

By adding an extra penalty term $\xi||\boldsymbol{Z}\boldsymbol{Z}^{\top} - \boldsymbol{I}||_F^2$ to the objective of problem (3.18), we can remove the orthogonal constraint. Denote $\boldsymbol{G} = \sum_{k=1}^{v} \frac{\partial \mathrm{DTW}_{\Omega}(\boldsymbol{Z}, \boldsymbol{Z}^{(k)})}{\partial \boldsymbol{Z}^{(k)}}$; then, the update rule for $\boldsymbol{Z}$ is as follows:

$$z_{i,j} \leftarrow z_{i,j} \frac{[4\xi \boldsymbol{Z}]_{i,j}}{[\boldsymbol{G} + 4\xi \boldsymbol{Z}\boldsymbol{Z}^{\top}\boldsymbol{Z}]_{i,j}}. \tag{3.19}$$

To guarantee the orthogonality of $\boldsymbol{Z}$, we set ξ as a relatively large value, $\xi = 10^6$, in our experiments. The derivation of equation (3.19) is given in Appendix D . Because we use the Karush–Kuhn–Tucker (KKT) condition to update Z, the objective value of GMSA can be ensured to decrease monotonically. However, note that the objective function is not jointly convex with regard to all the variables and that the alternating optimization algorithm is not guaranteed to converge to the global optimum. Therefore, a good initial guess can help the

Algorithm 2 : Alternating optimization algorithm for GMSA

Require: Input data sequences $\boldsymbol{X}^{(k)}$ for $k = 1, ..., v$.
Ensure: The optimal DNNs' parameters $\boldsymbol{\theta}^*$ and consensus sequence $\boldsymbol{Z}^*$.
1: **for** $k = 1, ..., v$ **do**
2: Run k-means on $\boldsymbol{X}^{(k)}$ to generate cluster indicator matrix $\boldsymbol{F}^{(k)}$;
3: Initialize $\boldsymbol{Z}^{(k)} = (\boldsymbol{F}^{(k)\top}\boldsymbol{F}^{(k)})^{-\frac{1}{2}}\boldsymbol{F}^{(k)\top}$;
4: Initialize $\boldsymbol{\theta}^{(k)} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}}||\boldsymbol{Z}^{(k)} - f_k(\boldsymbol{X}^{(k)}, \boldsymbol{\Theta})||_F^2$.
5: **end for**
6: **repeat**
7: Update the consensus $\boldsymbol{Z}$ using equation (3.19);
8: Update the DNNs' parameters $\boldsymbol{\theta}$ using a stochastic algorithm;
9: Recompute $\boldsymbol{Z}^{(k)} = f_k(\boldsymbol{X}^{(k)}, \boldsymbol{\theta}^{(k)})$ for $k = 1, ..., v$;
10: **until** convergence

algorithm to achieve a better optimal solution and converge faster. In this work, we separately run a k-means algorithm on the input sequences $\boldsymbol{X}^{(k)}$ to generate their cluster indicator matrices $\boldsymbol{F}^{(k)}$ for $k = 1, ..., v$. The initial value of $\boldsymbol{Z}^{(k)}$'s output of the DNNs are then computed using equation (3.14). Afterward, Algorithm 2 summarizes the alternating optimization procedure of GMSA.

3.6 Empirical Study

3.6.1 Compared methods

We compare GSCA and its variant GSCAEs with the following two-view baselines:

- Canonical time warping (CTW) [Zhou and Torre, 2009]—a direct combination of CCA and DTW;
- Canonical soft time warping (CSTW) [Kawano et al., 2019]—a probabilistic extension of CTW, where the alignment is considered a variable that follows Gibbs distribution. The alignment and projection matrices are alternatively optimized using the Expectation–Maximization (EM) algorithm.

- Autoencoder regularized CTW (AECTW) [Nie et al., 2016]—a variant of CTW with autoencoder-based regularizations;

- Deep CTW (DCTW) [Trigeorgis et al., 2018]—a direct combination of Deep CCA and DTW;

- Locally unsupervised manifold alignment (LUMA) [Wang and Mahadevan, 2009]—an unsupervised manifold alignment-based method that establishes a connection between any two samples from the two views by comparing their local geometries;

- Fuzzy granule manifold alignment (FGMA) [Li et al., 2020]—a variant of LUMA, where the local geometry information is collected in the fuzzy granule space instead of the original space.

- Generalized unsupervised manifold alignment (GUMA) [Cui et al., 2014]—another unsupervised manifold alignment-based method that encodes cross-view sample-wise correspondence into a binary matrix that is jointly optimized with the projections;

- Manifold alignment time warping (MATW) [Vu et al., 2012]—a hybrid method where sample alignment is performed by DTW and where projection matrices are optimized to preserve the underlying structures of the two views;

and compare GMSA with the following multi-view method:

- Generalized canonical time warping (GCTW) [Zhou and De la Torre, 2015]—Multi-set CCA is used to project data sequence into a shared subspace with an approximation of the DTW algorithm to align the projected sequences.

We note that DTW_Ω in the objective of our methods has different versions depending on the selection of the regularization functions $\Omega(\boldsymbol{\gamma})$. Therefore, we add suffixes -e and -s to our methods, indicating that $\Omega(\boldsymbol{\gamma})$ is Shannon entropy and squared ℓ_2 norm, respectively.

3.6.2 Evaluation measurements

All datasets in our experiments are divided into training, tuning, and test sets. We evaluate these methods by measuring class separation in the learned embedding spaces on the test set. First, we perform clustering tasks on the projections of the first view and evaluate how well the clusters agree with the ground-truth labels[2]. We follow the same procedure in [Wang et al., 2015a], where spectral clustering [Ng et al., 2001] is used to handle possibly non-convex cluster shapes. We set the number of clusters to the number of ground-truth classes available in each dataset. Clustering accuracy (ACC) and normalized mutual information (NMI) [Cai et al., 2005] are used as measurements for assessing the clustering performance. More details of their definitions are given in Appendix F. Second, we test the accuracy of a simple linear classifier on the learned embeddings. We train one-versus-one linear support vector machines (SVMs) [Chang and Lin, 2011] on the projected training set of the first view (label information is used). The trained model is then used to classify projections of the test set, and the percentage of errors is reported as a measurement of classification performance.

3.6.3 Parameter tuning

We select the optimal parameters that return the best evaluation measurement results on the tuning set for each method.

Two-view methods: Dimension d of the new subspace is selected from $\{5, 10, 20, 30, 50, 70\}$. For manifold alignment-based methods, we select the parameter that balances between sample matching and geometry preserving from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. The number of neighbors for building the k-NN graph that encodes the local geometry is selected from $\{1, 3, 5, 10, 15, 30\}$. We found that these methods return the best average results at $k = 5$. The trade-off parameter between the autoencoder regularization term and the alignment objective in AECTW and GSCAEs is selected using a grid search. For the soft whitening constraints in GSCA and GSCAEs, we set $\lambda_x = \lambda_y$, and their values are also selected using grid search. Another im-

[2]For GMSA, we use the projections of the views as their cluster indicators.

portant parameter is β, which controls the regularization in the smooth min operator. We set $\beta = 1$, as suggested in [Doan and Atsuhiro, 2019, Cuturi and Blondel, 2017]. For the DNNs used in the compared methods, their topology and configurations are data-dependent and are specified in the following subsections.

Multiview methods: For GCTW, similar to the two-view methods, we select a dimension of the new subspace from $\{5, 10, 20, 30, 50, 70\}$. To approximate the optimal warping paths, we use five hyperbolic tangent and five polynomial functions as the monotonic basics. The other parameters are set according to the original paper. In contrast to GCTW, our method GMSA projects the input sequences into the label space. Therefore, we choose the dimension d of the new space for GMSA such that it is identical to the number of classes available in the datasets. We use the same soft whitening regularization parameters for all the views: $\lambda_1 = \lambda_2 = ... = \lambda_v$, and the value is chosen using grid search. Let n_a, n_s, and n_l denote the average, shortest, and longest lengths, respectively, of the input data sequences. Then, the length of the consensus sequence Z is selected from a rounded set $\{n_s, \max(n_s, 0.5n_a), \max(n_s, 0.75n_a), n_a, \min(n_l, 1.25n_a),$
$\min(n_l, 1.5n_a), n_l\}$. The alternating optimization algorithm of GMSA is determined to be converged if the relative reduction of the objective is smaller than a tolerance $\epsilon = 10^{-5}$. In practice, we also terminate the algorithm if the number of iterations exceeds a prespecified value $iter\_max = 50$.

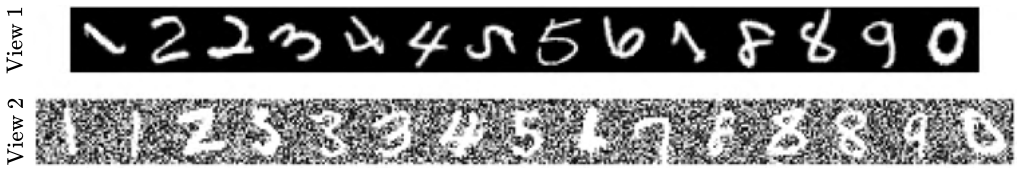
3.6.4 Two-view Data I: Noisy MNIST Digits

In this experiment, we utilize the MNIST dataset [LeCun et al., 1998], which consists of 28×28 grayscale digit $[0, 9]$ images divided into $60K/10K$ for training/testing. Following the procedure in [Wang et al., 2015a], we generate two-view data as follows. For the first view, we rescale the pixel to $[0, 1]$ and randomly rotate the images at angles uniformly sampled from $[-\frac{\pi}{4}, \frac{\pi}{4}]$. For each image of the first view, we randomly select an image of the same identity from the original dataset, add noise uniformly sampled from $[0, 1]$, and truncate the pixel value to $[0, 1]$. This image is further resized to 24×24 and used for the second view. $10K$ from $60K$ image pairs

of the original training set are set aside for tuning.

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X	M_1	M_2	I_2	M_3	M_4	I_4	M_5	I_5	M_6	M_7	M_8	I_8	M_9	M_{10}	
Y	M_1	I_1	M_2	M_3	I_3	I_3	M_4	M_5	M_6	M_7	M_8	I_8	I_8	M_9	M_{10}

(a)

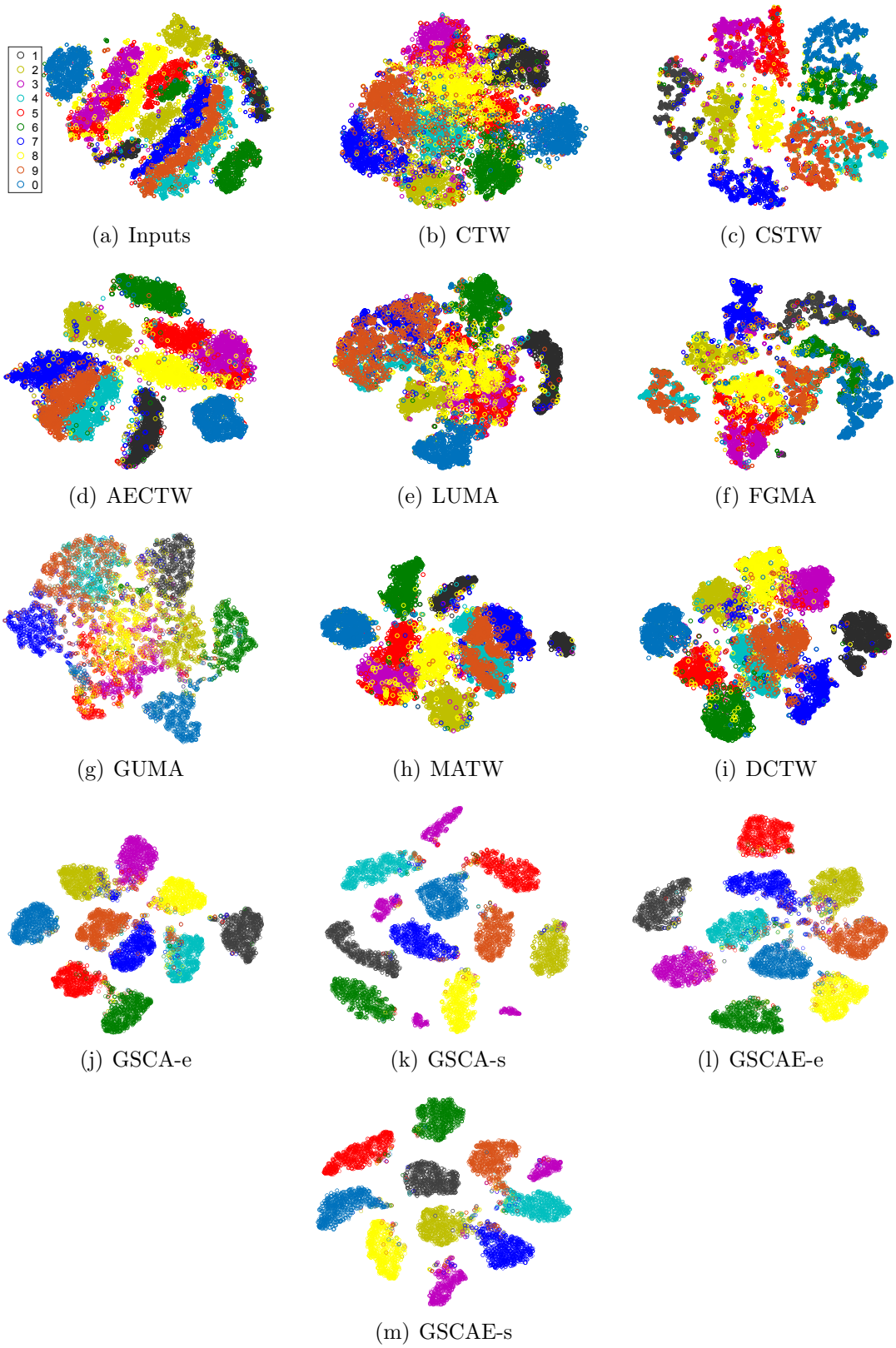


(b)

Figure 3.4: A toy example of how to generate misaligned sequences. (a) Hidden states generated by pHMM and (b) the corresponding two data sequences generated from the noisy MNIST digits dataset.

From $50K$ image pairs $\{(\boldsymbol{x}_i, \boldsymbol{y}_i, l_i)|1 \leq i \leq 5 \times 10^4, \boldsymbol{x}_i \in \mathbb{R}^{784}, \boldsymbol{y}_i \in \mathbb{R}^{576}, l_i \in \{0, ..., 9\}\}$ of the new training set, we generate two misaligned sequences using a profile hidden Markov model (pHMM) [Eddy, 1998]. Specifically, we generate two state sequences, consisting of `MATCHING` state M_i that emits the i^{th} matching sample and `INSERT` state I_i intended for emitting sample replication. The transition probability is chosen such that from any state, the next state is `MATCHING` with probability 0.6 and `INSERT` with probability 0.4. We terminate the state sequences after reaching the $(5 \times 10^4)^{\text{th}}$ `MATCHING`. The state M_i of the first sequence corresponds to sample $\boldsymbol{x}_i$. For state I_i, we replicate $\boldsymbol{x}_i$ by randomly selecting a sample $\boldsymbol{x}_{l=l_i}$ from its class (having the same identity). (Similarly for the second sequence). Figure 3.4 shows a toy example of how to generate misaligned sequences using pHMM for a given small set of 10 training image pairs.

In this experiment, we used feed-forward neural networks for all DNN-based models, including DCTW, GSCA, and GSCAEs. To parameterize the projection functions that map data from original spaces to the new subspace, we use two fully connected networks whose numbers of sigmoid units at hidden layers are $1200-1200-1200$ (for the first view) and $1000-1000-1000$ (for the second view). Note that each network includes a BN layer of d units on the top as an output layer. The view reconstruction in GSCAEs is performed by the symmetric DNNs. Figure 3.5 visualizes the first view in the original space and its projections in the subspaces



(a) Inputs (b) CTW (c) CSTW

(d) AECTW (e) LUMA (f) FGMA

(g) GUMA (h) MATW (i) DCTW

(j) GSCA-e (k) GSCA-s (l) GSCAE-e

(m) GSCAE-s

Figure 3.5: t-SNE [Van der Maaten and Hinton, 2008] visualization of the projected test set of noisy MNIST digits returned by different methods.

Method	ACC (%)	NMI (%)	Error (%)
Inputs	45.14 (5.27)	48.05 (6.77)	15.61 (2.86)
CTW	66.33 (4.47)	52.17 (4.81)	20.28 (3.21)
CSTW	75.16 (1.62)	73.91 (2.01)	7.92 (1.50)
AECTW	82.62 (2.22)	79.66 (2.12)	8.24 (1.72)
LUMA	62.37 (3.21)	61.25 (3.71)	26.23 (3.64)
FGMA	66.51 (2.19)	65.93 (2.87)	24.36 (2.92)
GUMA	60.38 (3.47)	54.25 (2.90)	28.03 (3.14)
MATW	69.18 (4.86)	67.44 (5.13)	21.39 (3.03)
DCTW	86.46 (1.13)	84.22 (2.09)	6.23 (1.46)
GSCA-e	95.14 (1.03)	93.14 (1.12)	**2.80 (0.61)**
GSCA-s	90.57 (0.95)	89.93 (1.17)	5.06 (0.92)
GSCAE-e	**96.67 (0.81)**	**95.66 (0.68)**	3.12 (1.02)
GSCAE-s	91.48 (1.15)	91.35 (0.87)	4.47 (1.33)

Table 3.1: Clustering (ACC, NMI) and classifying (Error) results on the noisy MNIST digits dataset. The data sequences are generated randomly five times using the pHMM-based procedure. Each method is performed on these data to learn the new embeddings and the average results along with variances on projections of the test set are reported.

learned by different methods. The class separation results are shown in Table 3.1.

The results show that among manifold alignment-based models, FGMA had better scores than LUMA and GUMA. FGMA evaluates the local geometry of the data after converting them into a fuzzy granule space. Thus, FGMA can discover more complex local structure information. MATW is a hybrid model. Differing from LUMA, FGMA, and GUMA, MATW discovers sample correspondences by DTW. Thus, it can take advantage of sequential order in the data to find the cross-view correspondence. Nevertheless, these models returned poor results on noisy MNIST digits datasets because the noise corrupted the geometric information. In contrast, the deep learning-based models returned much higher class separation results, even in noisy conditions. These methods mapped samples of the same class to similar locations while suppressing noise and rotational variation in the data. We also observe that models with a smooth approximation of DTW, including GSCA and GSCAEs, worked much better than CTW, AECTW, and DCTW, which directly combine the original DTW with variants of CCA. By minimizing the differentiable version of DTW, alignment and projection can be optimized in a unified manner. CSTW also implicitly optimizes DTW because it considers the warping path as a probabilistic variable. However, its EM algorithm still updates the alignment and projection matrices alternatively, which is also prone to sub-optimal solutions.

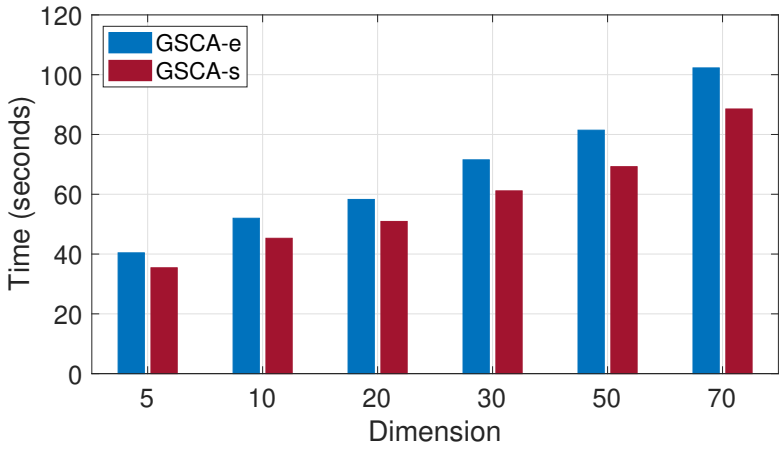


Figure 3.6: Average times for computing stochastic gradients of GSCA-e and GSCA-s over a batch size ratio $\alpha = 0.1$ (equivalent to a batch size of about $1.5K$) on noisy MNIST digits datasets with different dimensions d of the learned subspace.

The experimental results also show that by combining CTW or GSCA with autoencoders (forming the variants AECTW or GSCAEs) and carefully tuning the trade-off parameters their performances can be improved. We note that each method proposed in this chapter has two versions depending on the selection of the regularization $\Omega(\boldsymbol{\eta})$. Figure 3.6 shows the average times for computing the stochastic gradients of GSCA-e and GSCA-s over different dimensions d. Because of the sparsity of the gradient induced by squared ℓ_2 norm, training GSCA-s and GSCAE-s are generally faster than GSCA-e and GSCAE-e, respectively. However, this advantage comes at an expense of slightly lower class separation scores.

3.6.5 Two-view Data II: Acoustic and Articulatory Recordings

We next evaluate the performances of the two-view methods on the Wisconsin X-ray microbeam (XRMB) corpus [Westbury et al., 1994], which consists of 2537 utterances recorded from 47 American English speakers. Lengths of the utterances vary from 63 to 2941 frames. Each frame is basically described by $39D$ acoustic features (13-dimensional mel-frequency cepstral coefficients [MFCCs] along with their first and second derivatives) and $16D$ articulatory features (horizontal/vertical displacement of 8 pellets attached to the tongue, lips, and jaw). To incorporate context information and generate two sequential views of different frame rates, we slide windows of 7 and 9 frame sizes over each utterance with one frame step size. The frames

Method	ACC (%)	NMI (%)	Error (%)
Inputs	48.54 (3.51)	50.26 (3.65)	36.15 (4.51)
CTW	67.74 (3.11)	67.18 (3.32)	28.05 (3.88)
CSTW	68.82 (2.59)	67.49 (2.63)	27.64 (2.97)
AECTW	68.16 (2.32)	69.55 (2.69)	27.92 (3.63)
LUMA	56.23 (3.16)	56.83 (2.88)	30.33 (3.47)
FGMA	58.12 (2.92)	58.41 (2.95)	29.62 (3.03)
GUMA	55.73 (3.25)	54.61 (3.12)	30.97 (3.55)
MATW	64.51 (3.04)	63.38 (2.92)	29.35 (3.85)
DCTW	74.51 (3.07)	75.19 (2.90)	27.50 (3.28)
GSCA-e	82.45 (3.23)	82.83 (2.85)	26.48 (3.05)
GSCA-s	81.68 (2.84)	81.02 (3.19)	26.87 (2.94)
GSCAE-e	**83.90 (2.93)**	**82.91 (3.15)**	**25.35 (3.13)**
GSCAE-s	82.02 (3.02)	80.93 (2.89)	26.07 (2.86)

Table 3.2: Phone class separation on the projections of the acoustic view learned by different methods. The testing set is randomly divided into six folds. Clustering and classification tasks are performed on each fold and the average results along with their variances are reported.

within the windows are concatenated, resulting in $273D$ acoustic and $144D$ articulatory input samples. Because each original frame belongs to one of 41 phone classes, we consider the labels of the central frames as those of the newly generated inputs.

The utterances are presently characterized by two sequences whose lengths are different and the sample correspondences are also missing. We randomly divide them into 1415/471/471 for training/tuning/testing. We use RNNs for GSCA and GSCAEs to better capture the sequential nature of the data in this experiment. Specifically, for each view, we stack three LSTM units with the same numbers of memory cells (1500 for acoustic view and 1200 for articulatory view) along with a fully connected BN layer of d units at the output to parameterize the projection function. The view reconstruction in GSCAEs is performed by the symmetric deep LSTM networks. Note that while training these models using Algorithm 1, at each iteration, we randomly sample acoustic and articulatory input sequences that correspond to one of the 1415 training utterances to compute the stochastic gradient. Thereby, we can take better advantage of the sequential nature in the data for training the RNNs. For DCTW, we concatenate two views of the training utterances into two long sequences and feed them separately to two fully connected networks. These networks consist of three hidden layers whose activation functions are ReLU, and the numbers of the units are $1500 - 1500 - 1500$ for the acoustic view and

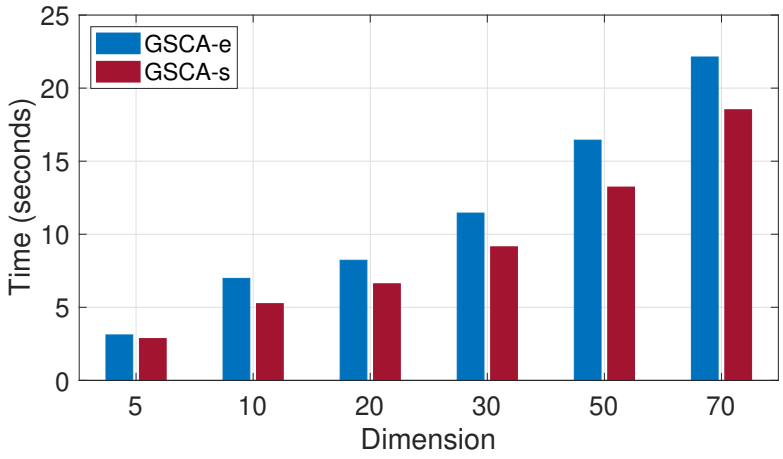


Figure 3.7: Average times for computing stochastic gradients of GSCA-e and GSCA-s on the XRMB dataset. For a fair comparison, computation times for backpropagation (or BTT) are excluded. The computation is taken on minibatches with the sizes are equal to average length of the training utterances (about $1K$ samples).

$1200 - 1200 - 1200$ for the articulatory view.

Table 3.2 shows the phone class separation results on representations obtained by different methods. Similar to the results on the noisy MNIST digits dataset, the DNN-based methods outperformed CTW, CSTW, AECTW, and the manifold alignment-based methods. The DNNs enable those methods to approximate projection functions nonlinearly, improving the quality of the learned embeddings. In this experiment, the designed RNNs have shown positive effects on our methods. They allow the models to better capture the sequential relationships among data samples. As a consequence, GSCA and GSCAEs achieved the highest scores among the compared methods. We also see that the results of AECTW and GSCAEs surpass those of CTW and GSCA, respectively. This again validates the benefits of coupling autoencoder-based regularizations with the objective functions for providing a better trade-off between view-specific and cross-view information.

We then investigate average times for computing stochastic gradients of GSCA-e and GSCA-s. Figure 3.7 shows that the computing gradients of GSCA-e and GSCA-s are very fast. This efficiency originates from the use of the new generalized smooth DTW. Note that, by setting $\Omega(\boldsymbol{\eta})$ in DTW_Ω to be squared ℓ_2 norm, the training time is further reduced as stochastic gradient GSCA-s is sparser than that of GSCA-e. However, as shown in Table 3.2, the class separation results were slightly decreased. Because $\text{DTW}_{\Omega=\text{squared } \ell_2}$ is a nonexact

approximation of DTW, GSCA-s and GSCAE-s included some certain bias in comparison with the Shannon entropy-based versions.

3.6.6 Multiview Data I: Human Actions with Multiple Feature Sets

In this experiment, we evaluate the performances of multiview methods, including GMSA and GCTW, on the Weizmann dataset [Gorelick et al., 2007], which consists of 90 videos of nine subjects, each performing ten actions: wave-one-hand (wave 1), wave-two-hand (wave 2), side, jump-in-place (pjump), jump-forward (jump), jack, skip, bend, walk, and run. Similar to [Hoai and De la Torre, 2014], we concatenate videos of the same subject into a long video sequence following the presented order of the actions. Each frame of these video sequences subtracts the background and is rescaled to the size 80×40. These types of features can be computed to characterize the frames, including *type 1:* binary, *type 2:* Euclidean distance transform [Maurer et al., 2003], and *type 3:* solution of the Poisson equation [Gorelick et al., 2006]. We generate three-view sequential data for training by selecting video sequences of the first three subjects, each of which is represented by one of the three feature types without repetition. As a result, each view of the data has different features, and each frame of the views belongs to one of the ten classes (see Figure 3.8 for more details). To reduce the dimensions of the feature space (3200), the top 123 principal components that preserve 99% of the total energy are selected. Videos of the next three subjects are used for tuning, and the remaining subjects' videos are utilized for testing.

For GMSA, we use RNNs to parameterize the projection functions. We use a similar network configuration for all the views because three views of the datasets have the same input dimensions. Specifically, the data of each view are passed through a deep network with three stacked LSTM units, each of which has 256 memory cells. The output layer of the network is a BN layer with $d = 10$ units. Because the new subspace in GMSA corresponds to the label space, we expect the learned representations can have cluster interpretability and better discriminability. Let $\boldsymbol{z}_i^{(k)} \in \mathbb{R}^d$ be the projection of the testing sample $\boldsymbol{x}_i^{(k)}$, we then assign $\boldsymbol{x}_i^{(k)}$ to class j such that $z_{j,i}^{(k)}$ is the largest element of $\boldsymbol{z}_i^{(k)}$. Table 3.3 shows the class separation

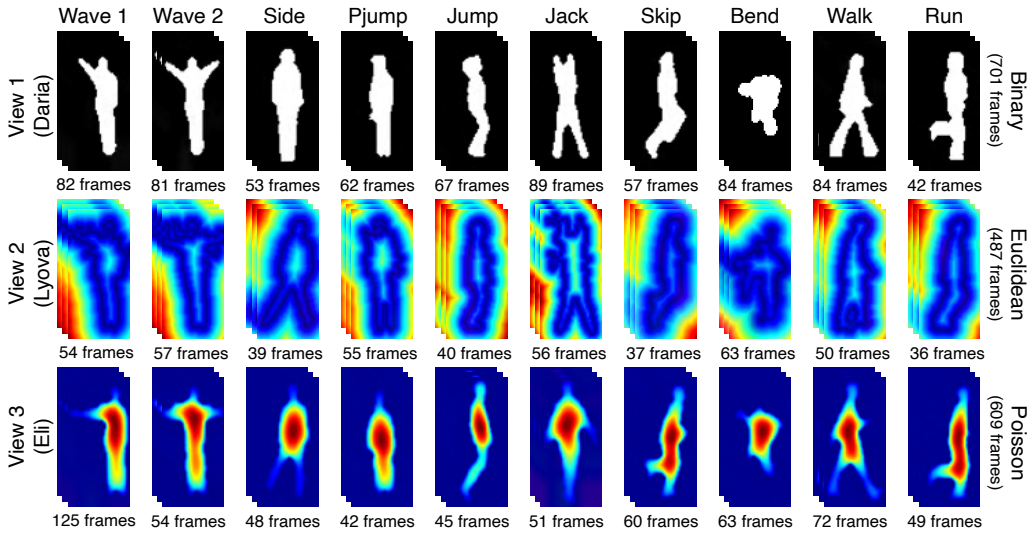


Figure 3.8: Three-view sequential data generated from the Weizmann dataset. The views are constructed by concatenating ten action videos of three subjects named Daria, Lyova, and Eli, respectively. Note that each view has different features: Binary (view 1), Euclidean distance transform (view 2), and solution of Poisson equation (view 3).

results on the representations learned by GCTW and GMSA.

Method		ACC (%)	NMI (%)	Error (%)
Input	view 1	53.75 (2.35)	62.48 (1.17)	14.21 (2.11)
	view 2	57.92 (1.72)	63.73 (2.02)	12.37 (1.27)
	view 3	49.15 (1.87)	55.22 (1.21)	13.53 (1.61)
GCTW	view 1	67.32 (2.06)	72.37 (1.27)	10.49 (1.14)
	view 2	68.49 (1.94)	74.01 (1.83)	8.72 (0.98)
	view 3	65.87 (1.55)	70.38 (2.13)	10.15 (1.36)
GMSA-e	view 1	86.65 (1.82)	88.50 (1.80)	**5.04 (1.85)**
	view 2	**88.03 (1.90)**	**90.02 (1.89)**	5.94 (1.85)
	view 3	84.62 (1.51)	86.12 (1.19)	6.56 (1.66)
GMSA-s	view 1	84.62 (2.08)	87.11 (1.96)	5.75 (1.91)
	view 2	87.59 (1.73)	88.54 (1.87)	7.04 (1.75)
	view 3	83.42 (1.57)	85.15 (1.49)	6.12 (1.55)

Table 3.3: Performance measures of clustering (ACC, NMI) and classifying (Error) on the projections of the Weizmann datasets, using GCTW and GMSA. Each method is run randomly five times, and their average results along with the variances on the test set are reported.

The results show that the performances of clustering and classification on the new embeddings returned by GCTW and GMSA are much better than those on the original space. These results indicate that integrating complementary information from different views helps the two methods to improve the quality of the new representations. In addition, the results also show that the improvement of GMSA is more considerable because it can learn more richer nonlinear embeddings. In contrast, GCTW limits itself to a shallow model where only lin-

ear projection matrices can be obtained. Another unfavorable property of GCTW is that the accuracy of its alignment procedure heavily depends on the selection of the monotonic basic functions. However, how to choose a suitable collection of basics for a particular dataset remains unclear. Therefore, the inappropriate settings of monotonic bases might degenerate the results of GCTW.

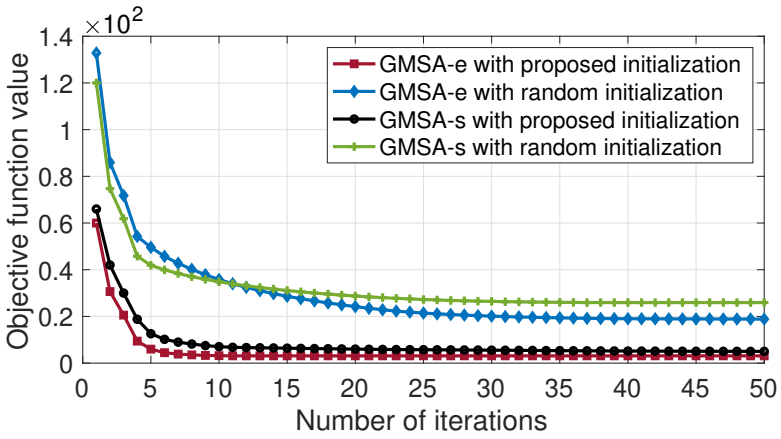


Figure 3.9: Convergence curves (objective function value averaged over five runs against the number of iterations) of GMSA-e and GMSA-s on the Weizmann dataset.

We then empirically explore the convergence of the optimization algorithm for GMSA. Each outer iteration of the algorithm includes two steps: finding the optimal consensus label sequence and updating the parameters for all DNN branches of the models. Figure 3.9 shows the convergence cures (objective function value against the number of iterations) with and without the proposed initialization (see Algorithm 2) on the Weizmann dataset. The results show that the algorithm still converges even with a random start. However, the proposed initialization improved the performance of the optimization procedure significantly. Not only does this help the algorithm to converge faster, but a good initial guess also allows better solutions to be obtained. These results again elucidate the efficiency of the GMSA model.

3.6.7 Multiview Data II: MMI Facial Action Units

We next exploit the MMI facial expression dataset [Pantic et al., 2005], which contains more than 2900 videos of 75 different subjects, each performing a particular combination of an action unit (AU). In our work, we focus on videos of AU12, which corresponds to a smile. These videos

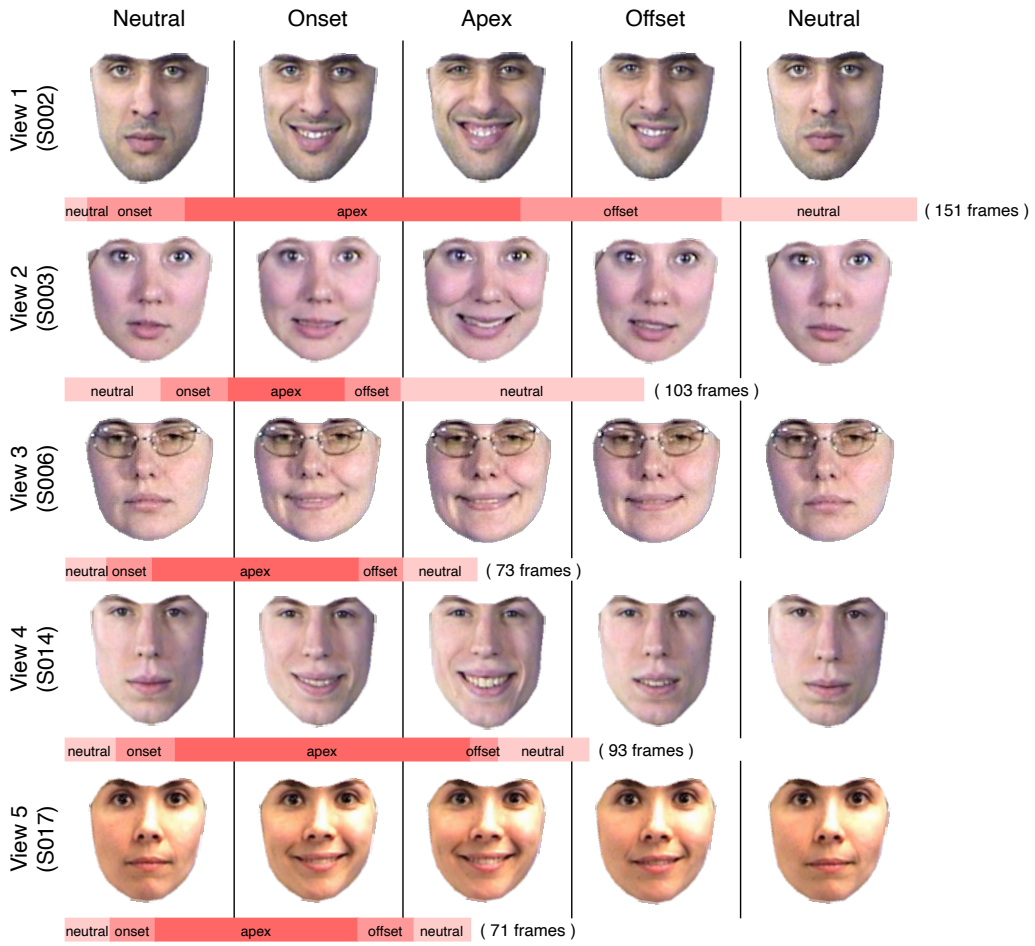


Figure 3.10: Five-view sequential data generated from the MMI facial expression dataset. The representative facial images of the classes are depicted. The bottom of each view shows the duration of the corresponding ground-truth temporal labels along with the total number of frames.

consist of different number of frames, and each belongs to one of three classes: *neutral* (when facial muscle is inactive), *apex* (when facial muscle intensity is strongest), and *onset* (when facial muscle starts to activate) or *offset* (when facial muscle begins to relax). We first preprocess each frame by performing face cropping and face alignment using dlib-ml [King, 2009]. The results are depicted in Figure 3.10. We then convert them to grayscale and reduce their dimension. Specifically, we utilize whitening PCA to pick the top 400 components, preserving 99% of the total energy. Finally, we generate sequential data with five views using videos S002–005, S003–023, S006–026, S014—009, and S017–004. Tuning and testing are performed on videos of the same subjects but in different trails (S002–006, S003–024, S006-025, S014–010, and S017–006).

In this experiment, RNNs are used to parameterize the projection functions for GMSA.

Method	ACC (%)		NMI (%)		Error (%)	
Input	64.40	(View 1)	61.51	(View 4)	10.29	(View 1)
GCTW	77.32	(View 1)	76.03	(View 4)	8.55	(View 5)
GMSA-e	**90.63**	(View 4)	**90.91**	(View 4)	**2.88**	(View 5)
GMSA-s	86.34	(View 4)	87.17	(View 4)	3.79	(View 3)

Table 3.4: Class separation results on the representations learned by GCTW and GMSA on the MMI facial expression dataset. Each method is run randomly five times, and their best average scores along with the corresponding views are reported.

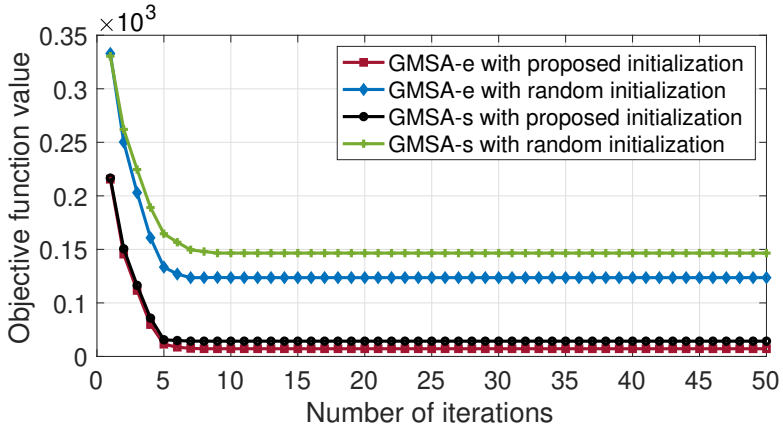


Figure 3.11: Convergence curves (objective function value averaged over five runs against the number of iterations) of GMSA-e and GMSA-s on the MMI facial expression dataset.

We stack two LSTM units, which each has 800 memory cells, and a BN layer with $d = 3$ units as the output layer to form a deep network for each view. The projections of the views are used to predict the cluster labels and perform the classification task. Table 3.4 shows the results of GCTW and GMSA.

The results show the same pattern as on the Weizmann dataset: *i.e.*, the representations learned by GCTW and GMSA significantly improve the performances of clustering and classification tasks in comparison with those on the original input features. Because GMSA-s and GMSA-e are nonlinear methods, their results are much better than those of GCTW. We also note that the multiple alignments in GMSA are simpler than that in GCTW because of the introduction of the consensus sequences. GCTW discovers the sample correspondences by instead performing pairwise alignment between every two views, while there are up to five views in this dataset. Therefore, more errors potentially occurred and propagated through update iterations in GCTW.

Finally, we investigated the convergence of Algorithm 2 on the MMI facial expression

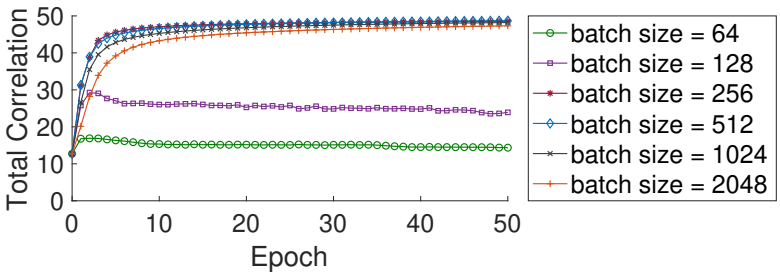


Figure 3.12: Learning curves (total correlation vs training epoch) of GSCA on MNIST dataset with d = 50.

dataset. Its convergence curve, which shows the objective value against the number of iterations, is depicted in Figure 3.11. The figure shows that the algorithm always converges, regardless of the initial conditions. Because the updated equation for the consensus label sequence satisfies the KKT conditions and the optimization for DNNs' parameters is based on the gradient descent method, the objective value is guaranteed to not increase after each iteration. In addition, as on the Weizmann dataset, we also observe that the proposed initialization significantly improves the performance of the algorithm. With a better initial value, the algorithm could converge with a much lower objective value, hence obtaining a superior optimal solution.

3.6.8 Stochastic optimization

In this subsection, we investigate the sensitivity of SGD algorithm with respect to the batch size on the learned models. Recall that, at each iterations, SGD randomly samples a subsequence from data of each view to estimate the stochastic gradient. Length of such subsequence is called batch size - an important parameter of the SGD algorithm. In general, this parameter is data-dependent and often selected using grid search. We, therefore, select the batch size from the set $\{64, 128, 256, 512, 1024, 2048\}$ on the MNIST dataset. After learning new representation for each view via GSCA model, we compute their total correlation in the shared space. The results are depicted in Figure 3.12.

We can observe that, when the batch-size is small (64 and 128), the algorithm converges quickly and the learned models induce low values of the total correlation. One possible reason is that the randomly sampled subsequences are too short that can not characterize well the

Dataset	Model	ACC (%)	NMI (%)	Error (%)
Weizmann	GMSA-e	88.03	90.02	5.04
	GMSA-e (w/o Binary view)	85.41 (↓ 2.62)	87.03 (↓ 2.99)	7.17 (↑ 2.13)
	GMSA-e (w/o Euclidean distance transform view)	83.26 (↓ 4.77)	85.82 (↓ 4.20)	6.49 (↑ 1.45)
	GMSA-e (w/o Solution of the Poisson equation view)	86.93 (↓ 1.10)	88.14 (↓ 1.88)	6.08 (↑ 1.04)
MMI	GMSA-e	90.63	90.91	2.88
	GMSA-e (w/o view 1)	87.42 (↓ 3.21)	87.67 (↓ 3.24)	3.91 (↑ 1.03)
	GMSA-e (w/o view 2)	89.05 (↓ 1.58)	88.93 (↓ 1.98)	3.35 (↑ 0.47)
	GMSA-e (w/o view 3)	89.17 (↓ 1.46)	88.62 (↓ 2.29)	3.28 (↑ 0.40)
	GMSA-e (w/o view 4)	85.56 (↓ 5.07)	85.15 (↓ 4.87)	5.11 (↑ 2.23)
	GMSA-e (w/o view 5)	88.01 (↓ 2.62)	87.84 (↓ 3.07)	4.03 (↑ 1.15)

Table 3.5: Ablation analysis of GMSA-e. The views are removed one by one, ablating one corresponding branch of DNN from the model. The best class separation scores of the ablated GMSA-e along with their differences to the results of the original one (full views) are reported.

statistical information of the data sequences. With large enough batch sizes (256) the SGD algorithm performs much better. However, as we further increase the batch size, the performance is not improved. This result empirically suggests that for a particular dataset there exists a certain level of batch size beyond that SGD algorithm performs stably. This level is crucial for reducing the training time and space requirement. A simple way to specify this level is to use grid search as already described in this subsection.

3.6.9 Ablation analysis of GMSA

In this section, we conducted ablation experiments to investigate the multiview effect in the GMSA-e model. For a v-views dataset[3], the GMSA-e model consists of v branches of DNNs, and each of which maps an input data sequence from one view into the shared label space. Following the same procedure in [Sha et al., 2018] and [Deng et al., 2019], we remove the branches of DNNs one by one and report the results in Table 3.5. The results show that some views are more important than others. For example, the absence of the Euclidean distance transform view in the Weizmann dataset or view 4 in the MMI dataset produces the most significant reduction in the results of the model. However, all of the views contribute more or less to the improvement of the model's performance. GMSA-e with full views has better separation scores than it does with view absence. This result again verifies the advantages of GMSA, which can handle multiple sequential views instantly.

[3] $v = 3$ for the Weizmann human action dataset and $v = 5$ in case of the MMI facial expression dataset.

3.7 Chapter Summary

Multiview sequential data pose many challenges to representation learning. In particular, the data sequences of different views are often unequal in size and sample-wise mismatching. Therefore, in this chapter, we introduced GSCA, a DNN-based method that can discover sample correspondence implicitly while learning representations. By using a generalized smooth DTW distance, which is a differentiable approximation of the original DTW, our model can be trained using a gradient descent-based algorithm, where the gradient can be computed efficiently in terms of both time and space. Our model can be easily extended to improve its learning performance. For instance, we added two DNNs to the GSCA model for view reconstructions, forming a new variant GSCAEs. The second model allows a trade-off between view-specific and cross-view relationships when learning the representations. Given more than two data sequences, it is obvious that both GSCA and GSCAEs are inapplicable. Hence, we further develop the third model called GMSA that can simultaneously handle multiple data sequences and learn more interpretable representations. Through extensive experimentation on different publicly available datasets, our methods were compared with various baselines. The results show that the performances of our methods surpass those of the competitors of all the datasets.

The results in this chapter again validate the advantages of differentiable models we claimed in Chapter 1. First, differentiable objectives allow parameters in our models to be obtained in an unified manner. Therefore, training GSCA and GSCAEs can avoid sub-optimal solutions and evaluation scores of both GSCA and GSCAEs surpass those of all exising models. Second, training our models is more efficient because the stochastic gradient descent algorithm can be applied. Time and memory complexities for computing the stochastic gradient are only $O(b^2)$, where $b \ll n$ in size of the mini-batch that is much smaller than length of the data sequence. In fact, without SGD it is even impossible to train GSCA and GSCAEs by a normal personal computer as the memory requirements for computing the full-batch gradients are 18.6 and 360.6 gigabytes on MNIST and XRMB datasets, respectively. Third, differentiable models are easy to be extended. GSCAEs and GMSA are examples of extended models based on GSCA.

Chapter 4

Differentiable Models for Segmentation of Sequences

Kernel segmentation aims at partitioning a data sequence into several non-overlapping segments that may have nonlinear and complex structures. In general, it is formulated as a discrete optimization problem with combinatorial constraints. 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 chapter, we take a differentiable approach to alleviate the aforementioned issues. First, we introduce a novel sigmoid-based regularization to smoothly approximate the combinatorial 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.

4.1 Introduction

Recently, there has been an increasing interest in research community for developing machine learning and data mining methods for sequential data. This is due to the exponential growing in number of collected data sequences in wide range of fields, including computer vision [Hoai and De la Torre, 2012, Hoai and De la Torre, 2014, Zhou et al., 2012], speech processing [Harchaoui et al., 2009b, Seichepine et al., 2014], finance [Lavielle and Teyssiere, 2007, Si and Yin, 2013], bio-informatics [Vert and Bleakley, 2010, Maidstone et al., 2017], climatology [Reeves et al., 2007, Verbesselt et al., 2010] and traffic monitoring [Lévy-Leduc et al., 2009, Lung-Yut-Fong et al., 2012]. The main problem associated with analysis of these sequences is that they consists of a huge number of data points. 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. From Bayesian point of view [Fearnhead, 2006, Chen and Gupta, 2011], this issue was addressed by developing parametric models, which serve as definitions of the intra-segment homogeneity. Although having promising performances in practice, this approach lacks versatility, require extensive data modelling knowledge and sophisticated numerical methods for training. In this chapter, we focus on kernel-based segmentation. This nonparametric approach has no assumption on the underlying process that generates the data. Thus, it can adapt easily to different types of data sequences. In addition, it enables segments with much more nonlinear and complex structures to be discovered.

For a fixed number of segments, [Harchaoui and Cappé, 2007] proposed an optimal algorithm based on dynamic programming (DP) for segmentation of data sequence in the features space, which is associated with 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 [Celisse et al., 2018] and greedy algorithm [Truong et al., 2019] 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 chapter, we take a different approach to alleviate the aforementioned issues. More precisely, we introduce a novel sigmoid-based regularization, which smoothly approximates the combinatorial constraints of the kernel 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 [Nocedal and Wright, 2006]. 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

[1] including time for computing the cost matrix in the feature space [Celisse et al., 2018]

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.

The remainder of this chapter is organized as follows: Section 4.2 briefly presents some background for the models proposed in this chapter. The sigmoid-based regularization and the corresponding segmentation model called *Kernel clustering with sigmoid-based regularization* (KCSR) are introduced in Section 4.3. The stochastic variant of KCSR is described in Section 4.4. After describing how to modify the sigmoid-based regularization to form an extension of KCSR that can simultaneously segment multiple data sequences in Section 4.5, we reports experimental results in Section 4.6. Section 4.7 summarizes the chapter.

4.2 Preliminary

4.2.1 Notations

Throughout this chapter, 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).

4.2.2 Balanced kernel $k-$means

As mentioned in 2.1.3, 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 $G_{j,i} = 1$ if $\boldsymbol{x}_i$ is assigned to the j^{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 the kernel $k-$means [Dhillon et al., 2004, De la Torre, 2012, Zass and Shashua, 2005] can be expressed as follows:

$$J_{KKM}(\boldsymbol{G}) = \operatorname{Tr}(\boldsymbol{L}\boldsymbol{K}), \tag{4.1}$$

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

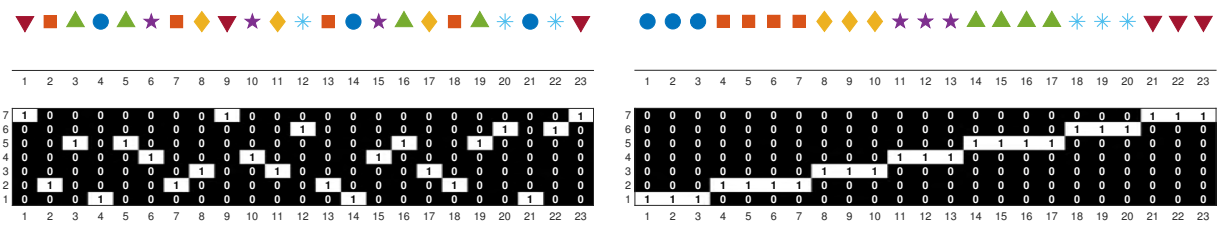
Kernel k-means is a strong approach for identifying clusters that are non-linearly separable in the original space. However, similar to its linear counterpart, kernel k-means is sensitive to outliers. More specifically, it often outputs unbalanced results that consists of too big and/or too small clusters under presents of anomaly data samples [Zhong and Ghosh, 2003]. To alleviate this issue, recently [Liu et al., 2017b] 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 (4.1) and the regularization to form a new objective of balanced kernel k-means

$$J_{BKKM}(\boldsymbol{G}) = \operatorname{Tr}(\boldsymbol{L}\boldsymbol{K}) + \lambda \operatorname{Tr}(\boldsymbol{G}\mathbf{1}\mathbf{1}^\top\boldsymbol{G}^\top), \tag{4.2}$$

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

4.3 Kernel Clustering with Sigmoid-based Regularization (KCSR)

Our intuitive idea is to reuse the robust objective of balanced kernel $k-$means (4.2) 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-



(a) Clustering task (b) Segmentation task

Figure 4.1: Toy examples of (a) Clustering task and (b) Segmentation task, where the given data and the corresponding indicator matrix are depicted.

segment indicator matrix must satisfy two constraints, including *boundary* and *monotonicity*, while the indicator matrix for clustering does not. This difference is illustrated Figure 4.1. 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 [Nocedal and Wright, 2006], quality of the proposed models' solutions is guaranteed.

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

$$f_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-\alpha(x-\beta)}}, \tag{4.3}$$

where β specifies the midpoint and α controls the steepness of the function curve at the midpoint. Figure 4.2 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

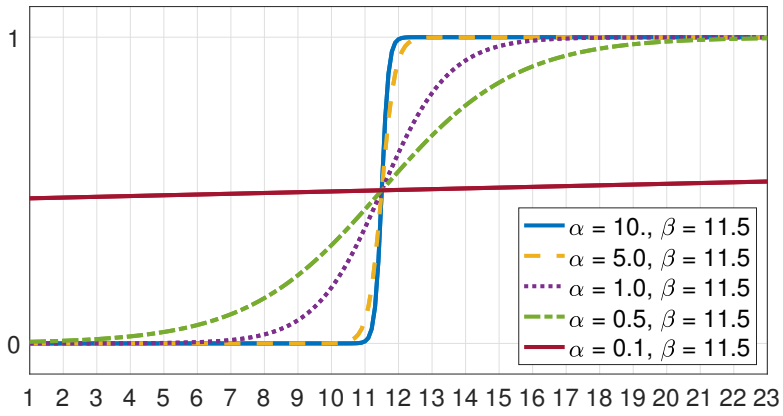


Figure 4.2: Sigmoid function with different values of the parameter α.

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 $\tau_i \in [1, 2]$ (continuously valued) as segment label of sample $\boldsymbol{x}_i$, then

$$\tau_i \approx 1 + f_{\text{sigmoid}}(i, \alpha, \beta). \tag{4.4}$$

For instance, if $\alpha = 10$ and $\beta = 11.5$, then $\tau_i \approx 1$ for $i < 11.5$ 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_j for $1 \leq j \leq k - 1$.

$$\tau_i \approx 1 + \sum_{j=1}^{k-1} f_{\text{sigmoid}}(i, \alpha, \beta_j). \tag{4.5}$$

Figure 4.3 illustrates an example of a summation of sigmoid functions defined in (4.5). Here,

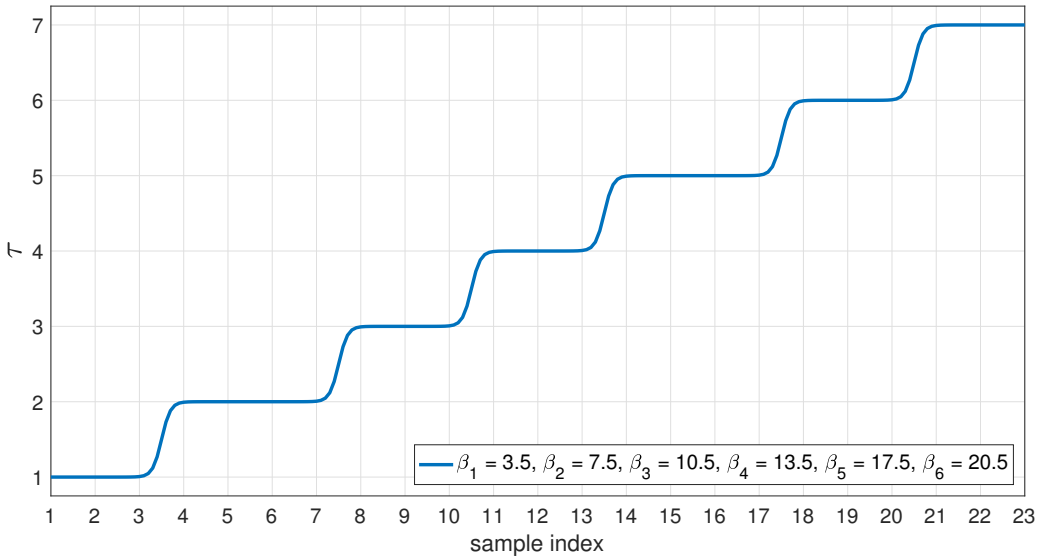


Figure 4.3: An example of the summation of sigmoid functions with a shared parameter $\alpha = 10$ and $k - 1$ different midpoint parameters $\beta_1, \ldots, \beta_{k-1}$, where $k = 7$.

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 $1 \leq \beta_1 < \ldots < \beta_{k-1} \leq n$ 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

$$\beta_j = \left(1 - \frac{\sum_{j\prime=1}^{j} e^{\gamma_{j'}}}{\sum_{j'=1}^{k} e^{\gamma_{j'}}}\right) + n \times \frac{\sum_{j\prime=1}^{j} e^{\gamma_{j'}}}{\sum_{j'=1}^{k} e^{\gamma_{j'}}}. \tag{4.6}$$

In equation (4.6), the ratio $\frac{\sum_{j\prime=1}^{j} e^{\gamma_{j'}}}{\sum_{j'=1}^{k} e^{\gamma_{j'}}}$ is in the range $[0, 1]$. Therefore, β_j always satisfies $1 \leq \beta_j \leq n$. In addition, the ratio becomes larger as j increases. This guarantees that $\beta_{j'} < \beta_j$ for $1 \leq j' < j \leq k - 1$.

It is notable that the summation of sigmoid functions in Figure 4.3 smoothly approximates the indicator matrix $\boldsymbol{G}$ of segmentation example in Figure 4.1(b). To make the observation more clear, we introduce the following approximation to each element of $\boldsymbol{G}$

$$G_{j,i} \approx \max\left(0, 1 - |\tau_i - j|\right). \tag{4.7}$$

This equation map the segment label τ_i 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. 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

$$\begin{aligned}
\underset{\gamma_1,\ldots,\gamma_k}{\operatorname{argmin}} \quad & \operatorname{Tr}\left(\boldsymbol{L}\boldsymbol{K}\right) + \lambda \operatorname{Tr}(\boldsymbol{G}\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{G}^\top) \\
\text{s.t.} \quad & \boldsymbol{L} = \boldsymbol{I}_n - \boldsymbol{G}^\top \left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1} \boldsymbol{G}, \\
& G_{j,i} = \max\left(0, 1 - |\tau_i - j|\right) \quad \forall j, i, \\
& \tau_i = 1 + \sum_{j=1}^{k-1} f_{\text{sigmoid}}(i, \alpha, \beta_j) \quad \forall i, \\
& \beta_j = \left(1 - \frac{\sum_{j\prime=1}^{j} e^{\gamma_{j'}}}{\sum_{j'=1}^{k} e^{\gamma_{j'}}}\right) + n \times \frac{\sum_{j\prime=1}^{j} e^{\gamma_{j'}}}{\sum_{j'=1}^{k} e^{\gamma_{j'}}} \quad \forall j.
\end{aligned} \tag{4.8}$$

Since $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_k]$ are unconstrained and continuous parameters, we can optimize objective function in (4.8) using the gradient descent algorithm. Let $J(\boldsymbol{\gamma})$ denotes the objective function

Algorithm 3 : 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 [**?**, **?**];
4: update $\boldsymbol{\gamma}_{(t+1)} = \boldsymbol{\gamma}_{(t)} - \eta\nabla\boldsymbol{\gamma}_{(t)}$;
5: **until** $|J(\boldsymbol{\gamma}_{(t+1)}) - J(\boldsymbol{\gamma}_{(t)})| \leq \epsilon$

in (4.8), then the gradient w.r.t parameters $\boldsymbol{\gamma}$ can be computed using chain rule.

$$\nabla\boldsymbol{\gamma} = \frac{\partial J(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}} = \frac{\partial J(\boldsymbol{\gamma})}{\partial \boldsymbol{G}} \times \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{\tau}} \times \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\beta}} \times \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\gamma}}, \tag{4.9}$$

where $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_n]$. More details on derivation of the gradient w.r.t $\boldsymbol{\gamma}$ is given in Appendix E. We call the proposed model *Kernel clustering with sigmoid-based regularization* (KCSR) and its optimization algorithm is given in Algorithm 3.

4.4 Stochastic KCSR

Kernel-based methods 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 $\boldsymbol{K}$, whose both time and memory complexities are of order $O(n^2)$. When n is large, these methods become computationally difficult. For example, length the MNIST sequence in the experimental section is $70K$. The corresponding kernel matrix $\boldsymbol{K}$ requires up to 36.5 GB for storage, which is definitely out of memory for a regular personal 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) [Robbins and Monro, 1951, Bottou, 1998, Spall, 2005]. SGD

Algorithm 4 : 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: $\eta = \eta_0 \times \rho^t$;
3: randomly sample a sub-sequence $\boldsymbol{X}_{(t)}$ of length b;
4: compute the partial kernel matrix $\boldsymbol{K}_{(t)}$;
5: 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: $\Delta\gamma_{(t)} = \eta\nabla\boldsymbol{\gamma} - \mu\Delta\gamma_{(t-1)}$;
7: $\boldsymbol{\gamma}_{(t)} = \boldsymbol{\gamma}_{(t-1)} + \Delta\gamma_{(t)}$;
8: **end for**

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

$$\nabla\boldsymbol{\gamma} = \frac{\partial J(\boldsymbol{\gamma})}{\partial \boldsymbol{G}_{(t)}} \times \frac{\partial \boldsymbol{G}_{(t)}}{\partial \boldsymbol{\tau}} \times \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\beta}} \times \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\gamma}}. \tag{4.10}$$

In equation (4.10), $\frac{\partial J(\boldsymbol{\gamma})}{\partial \boldsymbol{G}_{(t)}}$ 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 than computing the full-batch gradient as in equation (4.9). Details of the algorithm is given in Algorithm 4.

4.5 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

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

(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 [Qiu et al., 2019, Chang et al., 2019, Li and Todorovic, 2020].

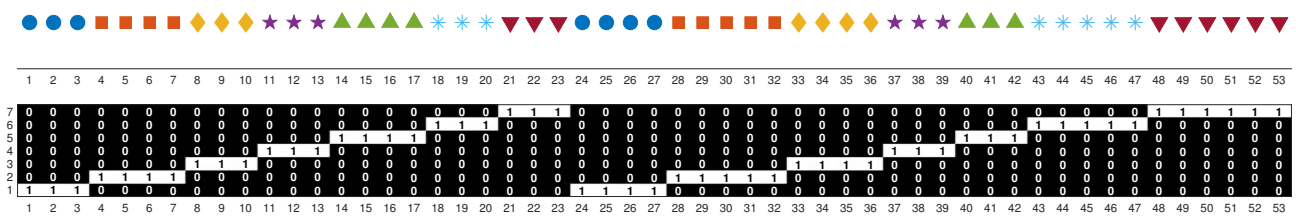
To solve the SSM problem, in this work, we introduce an extension of the proposed model termed *Multiple kernel clustering with sigmoid-based regularization* (MKSSR). MKSSR jointly partitions each data sequences into k segments such that the c^{th} segments of all the m sequences are matched [3]. Let $\boldsymbol{X}_p \in \mathbb{R}^{d\times n_p}$ 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 corresponding indicator matrix of $\boldsymbol{X}$. Similar to the original KCSR, each element of $\boldsymbol{G}$ is defined as in (4.7). However, in MKCSR, the segment label τ_i is computed as following

$$\begin{aligned}\tau_i = 1 + \sum_{j=1}^{m(k-1)} f_{\text{sigmoid}}(i, \alpha, \beta_j) \\ + (1-k)\sum_{p=1}^{m-1} f_{\text{sigmoid}}(i, \alpha, \sum_{q=1}^{p} n_q + 0.5).\end{aligned} \tag{4.11}$$

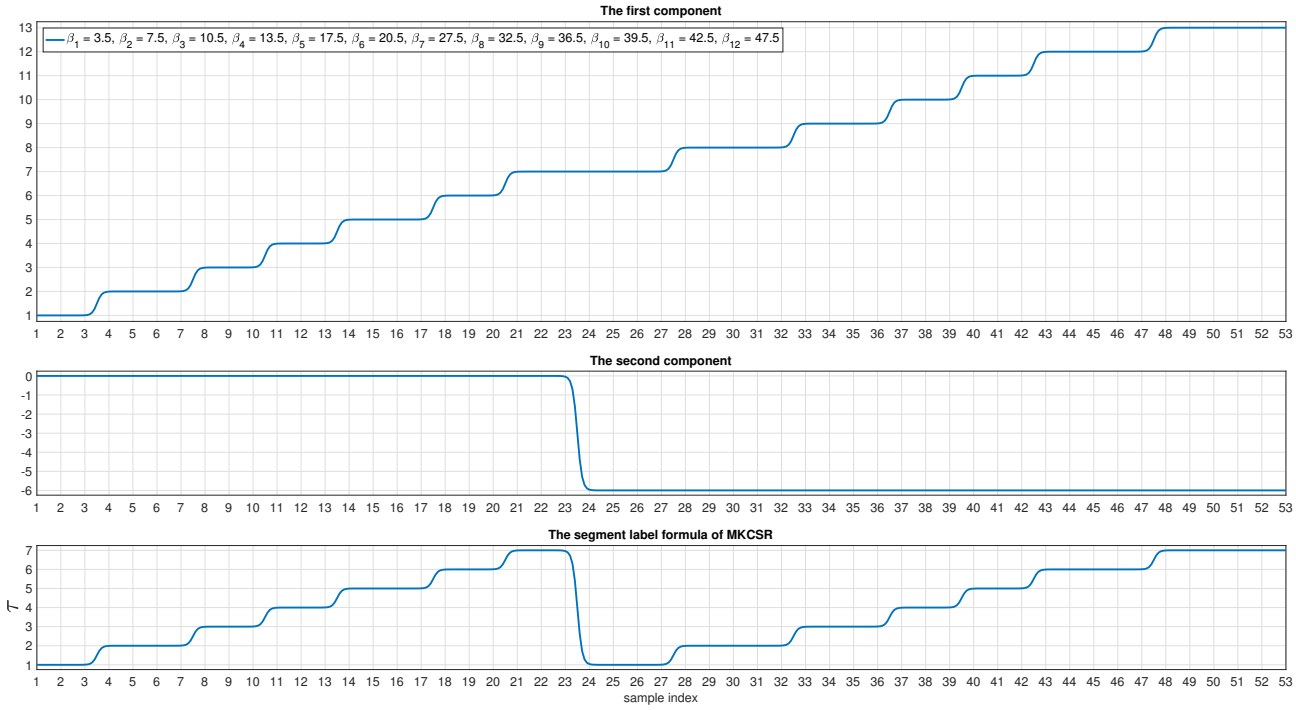
The function (4.11), which we call as cut-off summation of sigmoid functions, consists of two components. The first component is the summation of sigmoid functions. It plays a similar role as (4.5) in KCSR. The second component presents the cut-off points (a.k.a junction points), at which two among the 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 4.4.

The formulation (4.11) has $m(k-1)$ midpoint parameters, in which $\beta_{(p-1)(k-1)+1}, \ldots, \beta_{p(k-1)}$

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



(a)



(b)

Figure 4.4: 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.

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 $(p-1)(k-1)+1 \leq j \leq p(k-1)$

$$\beta_j = \left(1 + \sum_{q=1}^{p-1} n_q\right)\left(1 - \frac{\sum_{j'=(p-1)k+1}^{j} e^{\gamma_{j'}}}{\sum_{j'=(p-1)k+1}^{pk} e^{\gamma_{j'}}}\right) + \sum_{q=1}^{p} n_q \frac{\sum_{j'=(p-1)k+1}^{j} e^{\gamma_{j'}}}{\sum_{j'=(p-1)k+1}^{pk} e^{\gamma_{j'}}}. \tag{4.12}$$

By replacing the last two constraints in (4.8) with (4.11) and (4.12) 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.

4.6 Empirical Study

4.6.1 Compared methods

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

- Aligned clustering analysis (ACA) [Zhou et al., 2012] – a temporal clustering method that combines $k-$means with Dynamic time alignment kernel [Shimodaira et al., 2001].
- Ordered subspace clustering (OSC) [Wu et al., 2015] – a temporal clustering method that combines subspace clustering with linearly temporal regularization.
- Temporal subspace clustering (TSC) [Li et al., 2015a] – a temporal clustering method that combines subspace clustering with manifold-based temporal regularization.
- Approximate kernel segmentation (AKS) [Celisse et al., 2018] – 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) [Truong et al., 2019] – another heuristic approximation of the optimal kernel segmentation that detects the segment boundaries sequentially using the greedy algorithm.

4.6.2 Datasets

To evaluate performances of the above methods, we use a synthetic dataset and three 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 4.6(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

(a) Weizmann data

(b) Ordered MNIST data

Figure 4.5: (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.

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 4.6(b) for illustration.

Weizmann data. The Weizmann dataset [Gorelick et al., 2007] 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 [Hoai and De la Torre, 2014], 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 4.5(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.

Google spoken digits. Google's Speech Commands (GSC) [Warden, 2018] 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 Figure 4.8). 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) [Davis and Mermelstein, 1980] 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.

Ordered MNIST data. the MNIST dataset [LeCun et al., 1998] 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 [Zheng et al., 2021], 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 4.5(b)). Different from [Zheng et al., 2021], 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.

4.6.3 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. Therefore, we use accuracy and normalized mutual information [Cai et al., 2005], which can be used to evaluate both clustering and segmentation results, as evaluation metrics. The definitions of these measures are given in Appendix F.

4.6.4 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

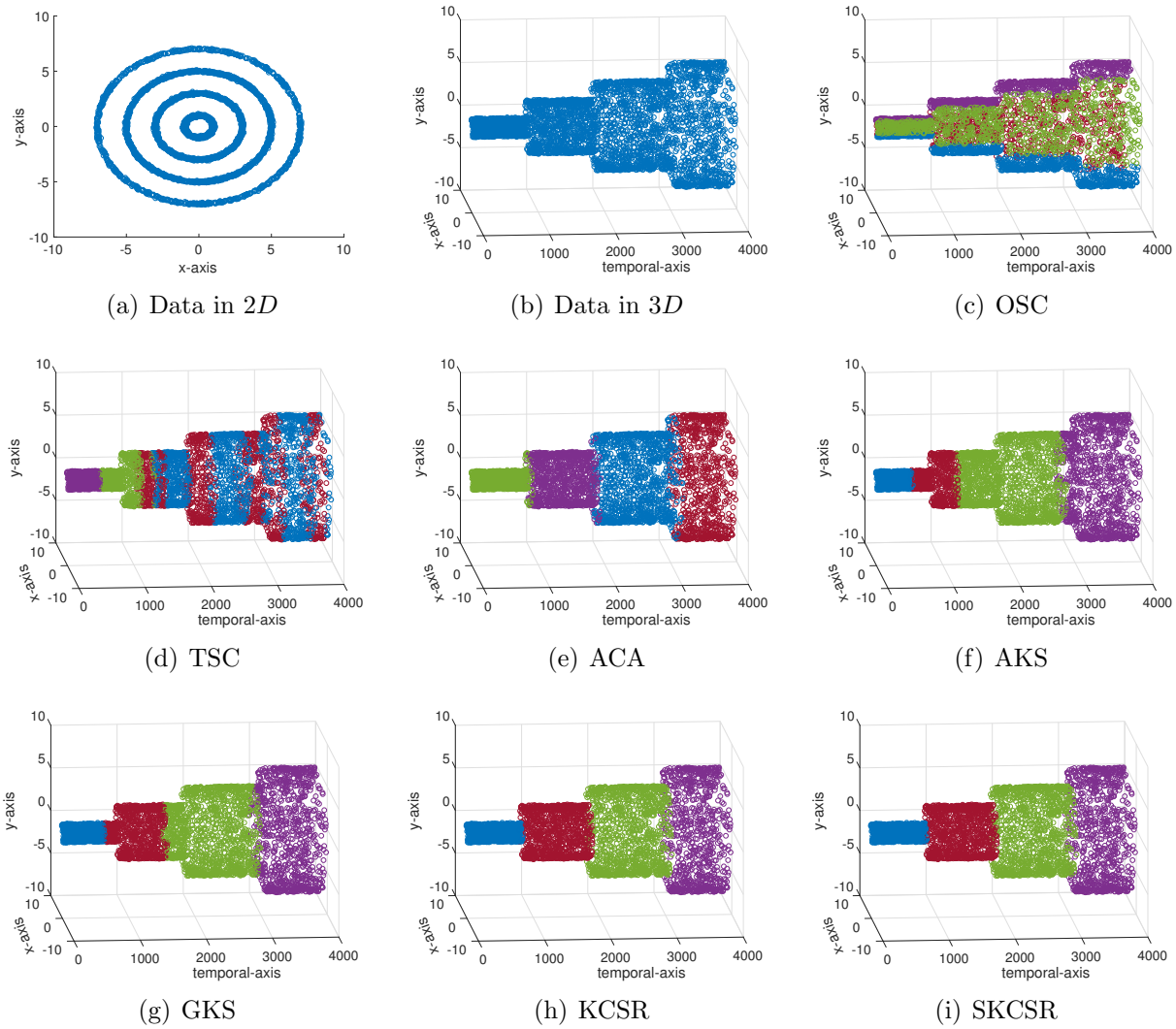
in the datasets. For ACA, its parameters nMa and nMi that specify the maximum and minimum lengths of each divided subsequence, respectively, are data-dependent. Let n be the sequence length, we select nMa from a rounded set $\{0.01n, 0.02n, 0.04n, 0.06n, 0.08n, 0.1n\}$ and set $nMi = \frac{nMa}{2}$. For temporal subspace clustering methods, including OSC 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 [4] 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).

4.6.5 Results discussion

Figure 4.6 visualizes the segmentation results on synthetic data and the evaluation scores are given in the first column of Table 4.1. We can observe that each segment of the generated data sequence has a circular structure. Therefore, the nonlinear regularization in sequential representation learning of OSC 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 OSC and TSC, where the regularization is just local[5], 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.

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

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



(a) Data in $2D$ (b) Data in $3D$ (c) OSC

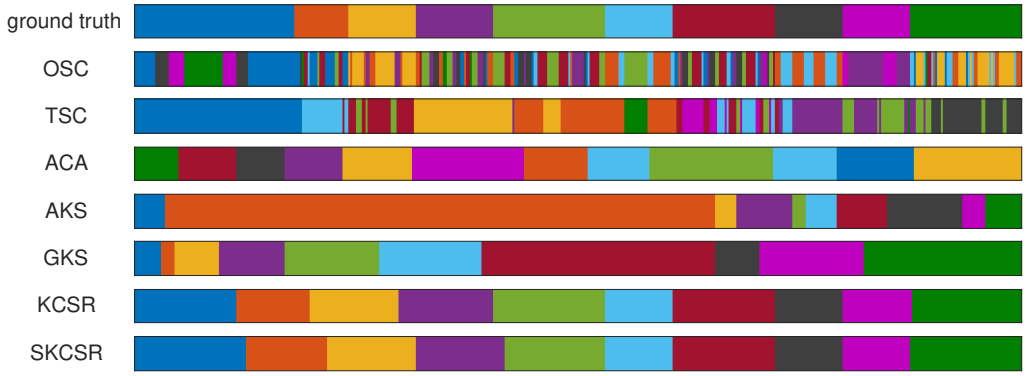
(d) TSC (e) ACA (f) AKS

(g) GKS (h) KCSR (i) SKCSR

Figure 4.6: 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.

Method	Synthetic data		Weizmann		Google		MNIST	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
OSC	0.0965 (0.0736)	0.1070 (0.0649)	0.4856 (0.0269)	0.4157 (0.0387)	0.2057 (0.0275)	0.1839 (0.0256)	–	–
TSC	0.4077 (0.0795)	0.3608 (0.0758)	0.7028 (0.0430)	0.7207 (0.0501)	0.6434 (0.0313)	0.6513 (0.0330)	–	–
ACA	0.9305 (0.0224)	0.9214 (0.0375)	0.7687 (0.0146)	0.7628 (0.0196)	0.8241 (0.0182)	0.7939 (0.0196)	–	–
AKCPD	0.6577 (0.0671)	0.6067 (0.0751)	0.7182 (0.0188)	0.7006 (0.0205)	0.6726 (0.0257)	0.6954 (0.0266)	0.6983 (0.0282)	0.7196 (0.0209)
GKCPD	0.6999 (0.0255)	0.7036 (0.0325)	0.5628 (0.0218)	0.6032 (0.0262)	0.7458 (0.0294)	0.7557 (0.0305)	–	–
KCSR	**0.9871** (0.0104)	**0.9959** (0.0024)	0.8835 (0.0092)	0.9071 (0.0107)	0.7914 (0.0172)	0.8109 (0.0154)	–	–
SKCSR	0.9870 (0.0092)	0.9847 (0.0077)	**0.8964** (0.0113)	**0.9151** (0.095)	**0.8826** (0.0165)	**0.9009** (0.0186)	**0.9681** (0.0155)	**0.9819** (0.0119)

Table 4.1: Segmentation results on four datasets, including synthetic data, Weizmann action sequences, noisy Google spoken digits and ordered MNIST data, returned by different methods. The mean score of each methods over five random runs along with its variance are reported.



(a) Weizmann

(b) MNIST

Figure 4.7: Visualization of segmentation results returned by the proposed methods and baselines on (a) Weizmann and (b) MNIST data. Different colors represent different clusters.

On the real-world data, including Weizmann action videos and Google spoken digits audio, the proposed models also outperformed the baselines. The corresponding segmentation results are visualized in Figure 4.7(a) and Figure 4.8, respectively, and the evaluation scores are shown in the second and third columns of Table 4.1. 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

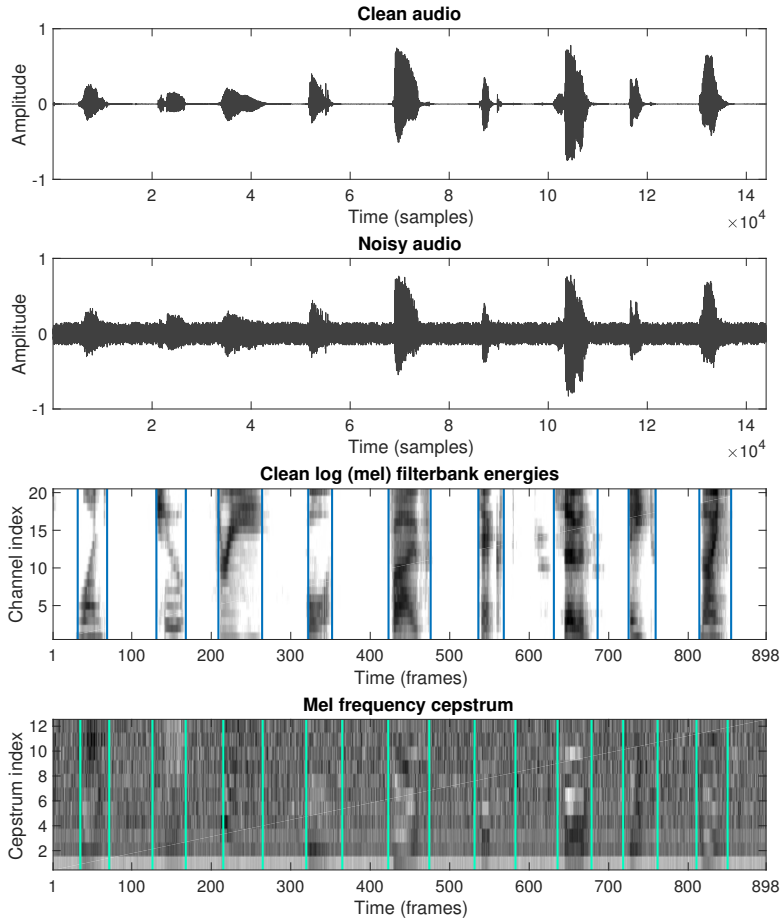
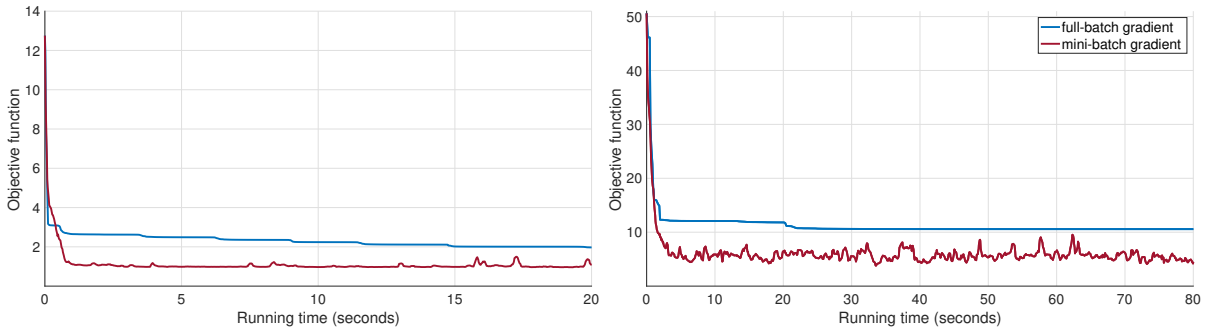


Figure 4.8: From the top to the bottom: clean audio of spoken digits [1,9], the audio contaminated by white noise, log filter-bank energies of the clean audio used for manual annotation (blue lines depict ground truth segment boundaries) and Mel-frequency cepstrum of the noisy audio (vertical lines show the midpoints βs of the mixture of sigmoid functions returned by SKCSR).

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.

On these two datasets, we also observe that evaluation scores of SKCSR are greater than those of KCSR. Thus, we further investigate convergence curves of these models, which are depicted in Figure 4.9. 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 [Bottou, 1998]. Therefore, SKCSR is more robust to noise in the data and



(a) Weizmann (b) Google

Figure 4.9: 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.

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 data. This data consists of $70K$ samples that makes implementation of these 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 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 ordered MNIST data in Figure 4.7(b) and the evaluation scores in the fourth column of Table 4.1 validate the advantages of SKCSR.

4.6.6 Evaluation of Multi KCSR

In this subsection, we evaluate performance of MKCSR – an extension of KCSR for handling multiple data sequences. We utilize concatenated Weizmann action videos in this experiment. 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. We compare MKCSR with temporal clustering methods, including OSC, TSC and ACA. For all the compared methods, we set the number of clusters $k = 10$ and select the other parameters following the same scheme as mentioned in subsection 4.6.4.

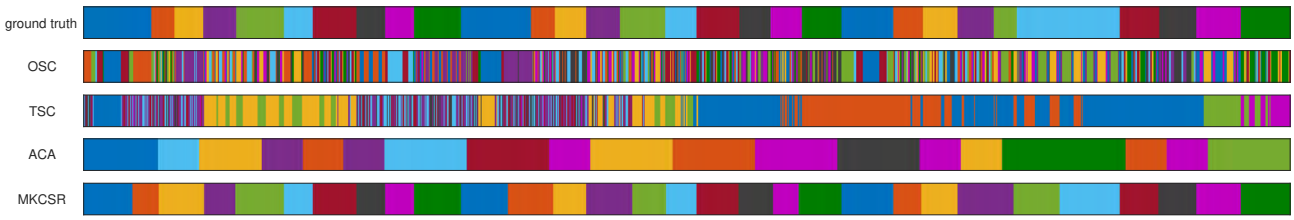


Figure 4.10: Visualization of segmentation results of OSC, TSC, ACA and MKCSR on three concatenated action video sequences from Weizmann dataset.

Method	ACC	NMI
OSC	0.1496 (0.0167)	0.1439 (0.0144)
TSC	0.1967 (0.0121)	0.2159 (0.0136)
ACA	0.5743 (0.0127)	0.5655 (0.0106)
MKCSR	**0.8509** (0.0139)	**0.8732** (0.0150)

Table 4.2: Segmentation results on concatenated action video sequences of three subjects from Weizmann dataset returned by different methods. The mean score of each methods over five random runs along with its variance are reported.

Figure 4.10 visualizes the segmentation results on multiple data sequences and Table 4.2 shows the corresponding evaluation scores. 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 column of Table 4.1), evaluation scores of OSC, 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.

4.6.7 Stochastic optimization

Similar to experiments for differentiable models for multi-view sequential learning, in this subsection, we also investigate the sensitivity of SGD algorithm with respect to the batch size when optimizing parameters for the SKCSR model. We perform the grid search on the noisy Google spoken digits dataset. The batch size is selected from the set $\{16, 32, 64, 128, 256, 512\}$. The optimization curves for different values of the batch size are illustrated in Figure 4.11.

We can observe that with small batch size (16 and 32) the SGD algorithm performs unsta-

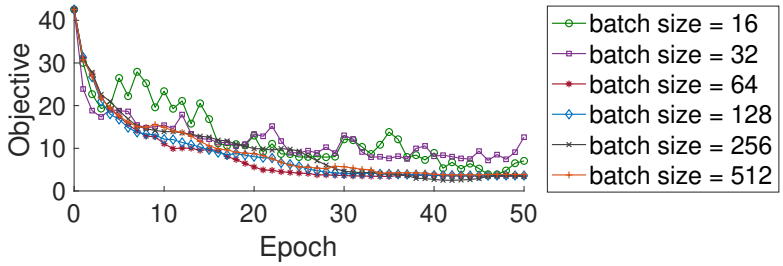


Figure 4.11: Optimization curves (objective vs optimization epoch) of SKCSR on Google spoken digits dataset.

bly and seems to need more time to converge. We note that, in our experiments, Google spoken digits dataset is heavily contaminated by noise. Therefore, small batch size might induce noisy estimation of the stochastic gradient making the algorithm to perform poorly. As the batch size increases, the algorithm converges faster and achieves better optimal solutions. The reason would be long subsequence contains more clean information that helps the algorithm to suppress noise when estimating the stochastic gradient.

4.7 Chapter Summary

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 chapter, we proposed KCSR to alleviate the aforementioned issues. 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.

Chapter 5

Application on Vehicle Detection

In this section, we describe our previous works on detecting vehicles from acceleration signals. We then discuss limitations of the current method and propose a new framework that exploits GSCAEs to detect vehicles from acceleration signals.

5.1 Current Vehicle Detection Method

Our previous works [Doan and Takasu, 2017, Doan and Takasu, 2019] studied a method for detecting vehicles on acceleration signals collected from bridge monitoring systems. More specifically, accelerometers are bonded to bridges to sense their vibration. When a vehicle approaches the position of an accelerometer, it induces vibration that is then measured by the sensor. The amplitude of the vibration increases as the vehicle approaches the accelerometer and reaches its highest magnitude when the vehicle is closest to the sensor. As the vehicle moves away from that position, the vibration damps. In ideal conditions, the signal generated by the accelerometer should oscillate as depicted in Figure 5.1 (a). However, sensors, especially those based on micro-electro-mechanical systems (MEMs) technology, are sensitive to environmental disturbances such as shocks and temperature changes. These sources of noise often produce high spikes and drift, adding nonoscillatory components to the signal. As a result, the shape of the signal is significantly distorted, impeding accurate vehicle detection. A real-world signal

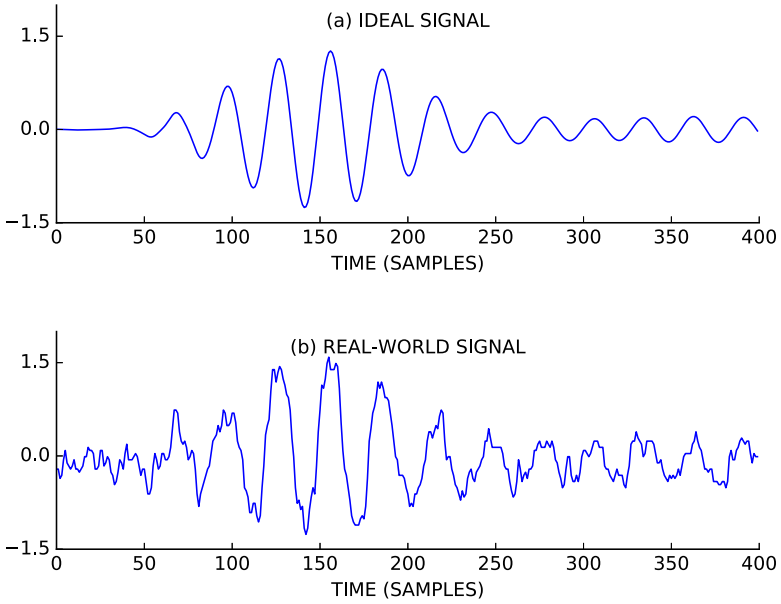


Figure 5.1: Ideal and real acceleration signals for a passing vehicle.

corresponding to a passing vehicle is shown in Figure 5.1 (b).

In [Doan and Takasu, 2017, Doan and Takasu, 2019], we constructed a method that consists of three phases, including noise reduction, oscillation detection, alignment-based fine tuning, to tackle the aforementioned difficulties. The diagram of the proposed method are illustrated in Figure 5.2.

Figure 5.2: Diagram of the vehicle detection method from acceleration signal.

Noise reduction. As the raw signal has an oscillatory component, which corresponds to passing vehicles, and a nonoscillatory component induced by environmental disturbances, the noise reduction method should be able to separate these components. In this step, we apply morphological component analysis (MCA) [Starck et al., 2005a, Elad et al., 2005, Starck et al., 2005b, Fadili and Starck, 2005] to perform this separation. Note that, MCA can only be applicable if each component has a sparse representation. Recently, [Selesnick, 2011b] introduced the tunable Q-factor wavelet transform (TQWT), which allows control of the oscillation of the wavelet function by tuning the Q factor. [Selesnick, 2011a] showed that the high Q-factor wavelet

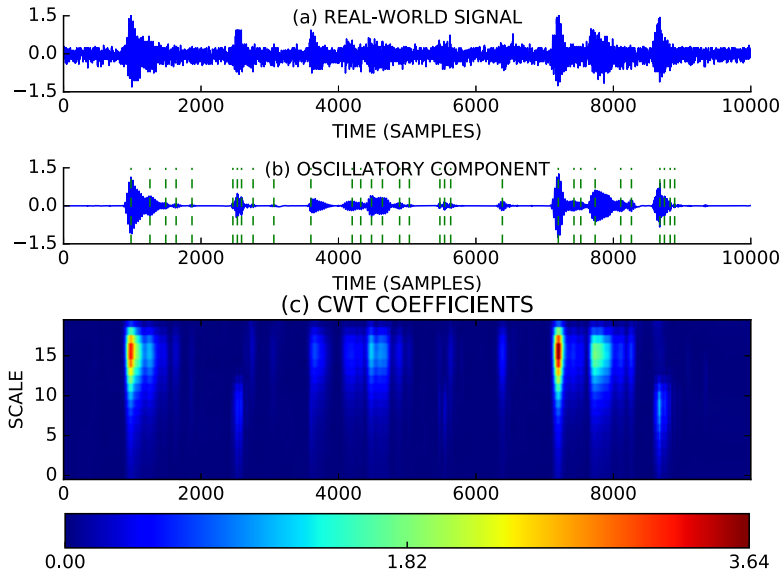


Figure 5.3: (a) Real-world signal, (b) its oscillatory component, and (c) CWT coefficients of the oscillatory component. The green dash lines show the detected oscillations after performing local search on the 2D wavelet coefficient matrix.

transform can represent oscillations sparsely, while the nonoscillatory signal can be sparsely represented by a low Q-factor wavelet transform. In addition, TQWT and its inverse TQWT^{-1} have low computation costs that are suitable for real-time applications. Therefore, we utilize TQWT to perform noise reduction.

Oscillation detection. The second phase takes a clean signal that is expected to contain only the oscillatory components as input. It then applies the wavelet transform with the Morlet wavelet as the mother wavelet function on this signal, resulting in a 2D wavelet coefficient matrix. Each oscillation is characterized by only one highest peak or deepest valley. Therefore, we take element-wise absolute values of the coefficient matrix and perform the local search procedure [Du et al., 2006] to detect the oscillations. An example is illustrated in Figure 5.3.

Refinement. In practice, there are often more than one accelerometers that are attached to the bridge at different positions. Information from these accelerometers can be used together to refine the results. More specifically, after performing the first two phases on each collected acceleration signal independently, sequences of epoches $\boldsymbol{t}^{(k)} = t_1^{(k)}, t_2^{(k)}, \ldots, t_{M^{(k)}}^{(k)}$, where $k = \overline{1, K}$ are indexes of the accelerometers and $M^{(k)}$ is number of detected oscillations, is generated. We observed that each true epoch in one sequence should have a corresponding epoch in each of the other sequences with an appropriate time lag. Therefore, this phase aligns the sequences

to find tuples of K matching epochs, which identify detected vehicles.

5.2 Limitations of The Current Method

The method described above still has some limitations. First, in the noise reduction phase, this method assumes that the oscillatory component merely corresponds to the vibration induced by passing vehicles and removes only nonoscillatory component, which is supposed to be generated by environmental disturbances. However, this assumption is not always correct in practice. For examples, white noise and seismic influences also result in vibration that are included in the signal measured by the accelerometers. Therefore, excluding nonoscillatory component can not completely eliminate all the noise in the signal. Second, the current method performs noise reduction on each individual signal collected from a single accelerometer. It ignores the complementary information of signals collected from the other sensors. Thus, the denoised signals tend to have low robustness and generalization ability because noise presents at the accelerometers are probably dissimilar from each other. Finally, the current method detects vehicles from the oscillatory component based on local search procedure [Du et al., 2006]. However, this procedure is originally proposed to detect peaks in the signal. Therefore, it might not have optimal performance on detecting oscillations.

5.3 Application of GSCAEs on Vehicle Detection

In this section, we propose a novel framework for vehicles detection from acceleration signals. By employing GSCAEs models as a core element, the new framework can alleviate the aforementioned issues of the current approach. Diagram of the proposed framework is depicted in Figure 5.4.

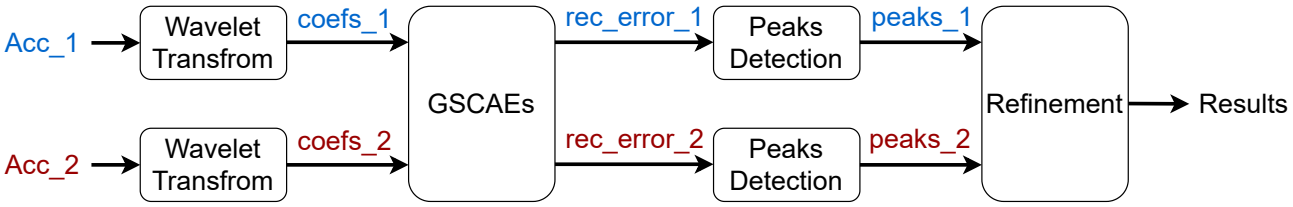


Figure 5.4: Diagram of the proposed framework.

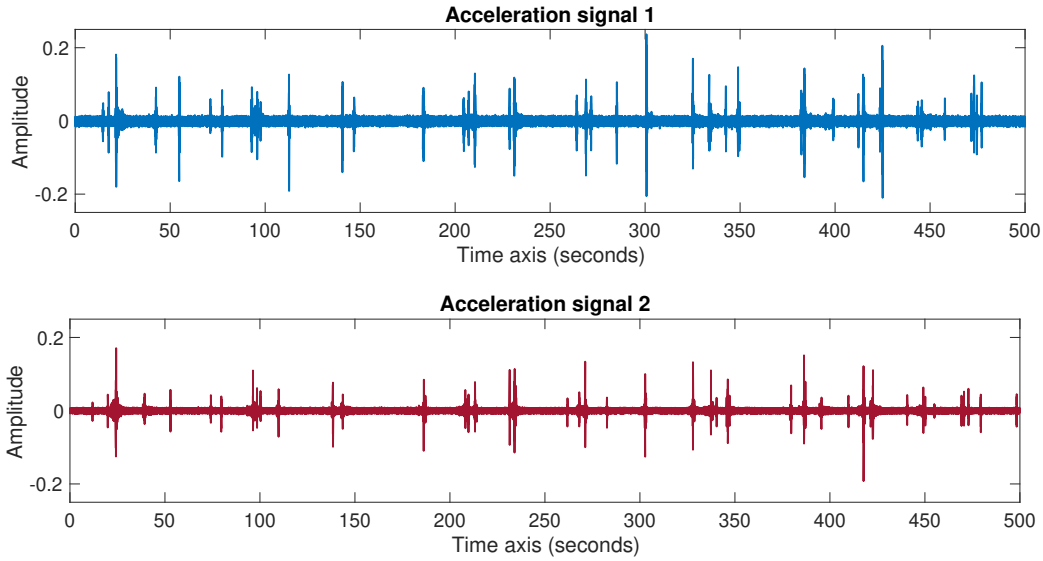


Figure 5.5: The raw acceleration signals collected from two different sensors.

5.3.1 The proposed framework

Our framework receives raw signals (Acc_1 and Acc_2) collected from the two different accelerometers as inputs. In general, these signals are contaminated by various types of noise. An illustration of the signals are shown in Figure 5.5. The framework applies wavelet transform on the input signals. Results of this transformation are two coefficient matrices $\boldsymbol{C}^{(1)} \in \mathbb{R}^{s_1 \times n_1}$ and $\boldsymbol{C}^{(2)} \in \mathbb{R}^{s_2 \times n_2}$ (coefs_1 and coefs_2), where n_1 and n_2 are the signal lengths and s_1 and s_2 are the number of wavelet scales (dimensions), respectively. Our framework can process two signals of different lengths and perform the wavelet transform at different scales. However, for simplicity, we henceforth assume that $n = n_1 = n_2$ and $s = s_1 = s_2$. Note that each element of $\boldsymbol{C}^{(1)}$ and $\boldsymbol{C}^{(2)}$ is a complex number. Thus, we extract modulus of the two matrices and denote them as $|\boldsymbol{C}^{(1)}|$ and $|\boldsymbol{C}^{(2)}|$, respectively.

The two matrices $|\boldsymbol{C}^{(1)}|$ and $|\boldsymbol{C}^{(2)}|$ are then fed to the GSCAEs model. Their corre-

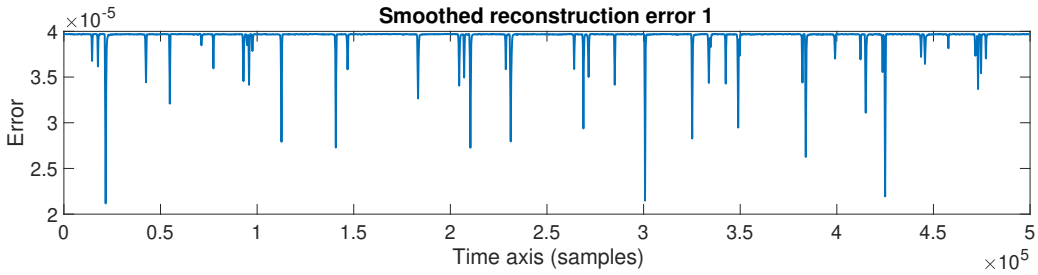


Figure 5.6: An example of reconstruction errors computed from wavelet coefficient matrix of the first acceleration signal.

sponding outputs are denoted as $|\widehat{\boldsymbol{C}}^{(1)}|$ and $|\widehat{\boldsymbol{C}}^{(2)}|$, respectively. The reconstruction errors $\boldsymbol{e}^{(v)} = [e_1, \ldots, e_n]$ for $v \in \{1, 2\}$ (rec_error_1 and rec_error_2) are computed using the following formulation

$$e_i^{(v)} = \sum_{j=1}^{s} \left(|\widehat{c}_{j,i}^{(v)}| - |c_{j,i}^{(v)}| \right)^2 \qquad i = 1, \ldots, n. \tag{5.1}$$

The reconstruction errors can be used as indicators to discriminate noise from clean data samples. The idea [Xia et al., 2015, Zhou and Paffenroth, 2017, Kieu et al., 2018, Kieu et al., 2019] is that the input data is reconstructed from a compressed hidden representation. Since this representation is very compact, it is only possible to reconstruct normal features from the input data. Therefore, the reconstruction errors for clean data are supposed to be much lower than those corresponds to noisy samples. An illustration of reconstruction errors is depicted in Figure 5.6.

We can observe that after smoothing the reconstruction error sequence form negative peaks at positions, where strong vibrations occur in the original signal. From this observation, our framework performs peaks detection on the reconstruction errors using the local search procedure [Du et al., 2006] as already mentioned in the previous section. After peak detection, we obtain two sequences of peaks denoted as peaks_1 and peaks_2, respectively, from two signals. We have a further observation that one correct peak in one sequence should have one corresponding peak in the other sequence with an appropriate time lag. Therefore, we match the peaks between the two sequences in a refinement step. The matching procedure is as follows:

- First, we specify the highest and lowest velocities to calculate the possible maximum and

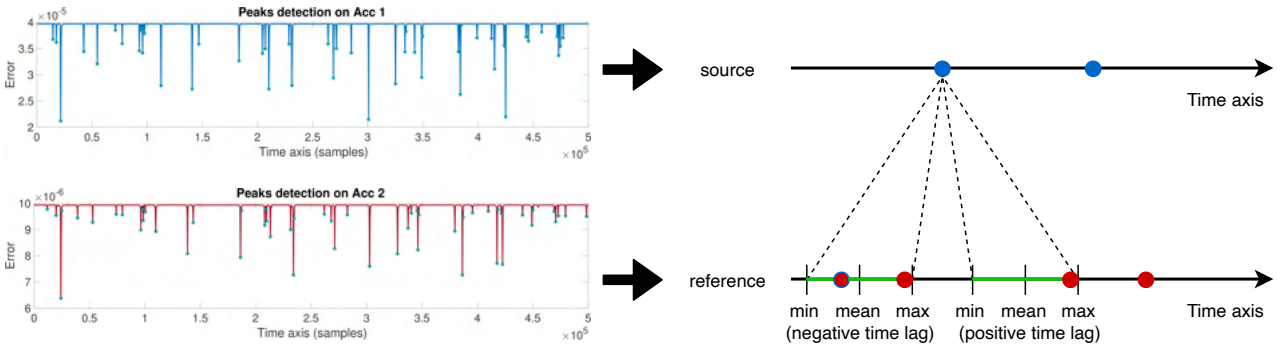


Figure 5.7: Illustration of peaks matching procedure in refinement of the proposed framework.

minimum time lags for a matching pair of peaks.

- Second, we then select the sequence with a smaller number of peaks as the "source". The other sequence will be the "reference".

- Third, the peaks in the source will be processed on-by-one. The corresponding peak in the reference sequence must satisfy the time lag condition. If there are more than one valid peak, the peak that is close to the mean of the time lag will be chosen. Once a peak in the reference sequence is selected, it will be excluded from the search for the next peak in the source sequence.

Figure 5.7 illustrates the matching procedure. Detected peaks (blue circles) in the source sequence are matched with their corresponding peaks (red circles) in the reference sequence. The green areas on time axis of the reference sequence depict the valid time lags for a particular peak in the source sequence. If there are more than one peaks of the reference lie in this areas, the peak that is close to the mean time lag will be selected. After refinement, we consider each matching pair of peaks as a detected vehicle. We expect that combination the detected results from the two sensors will further improve accuracy of the vehicle detection framework.

5.3.2 Empirical study

Data. We evaluate performance of the proposed framework on real-world acceleration signals collected from sensors attached on a prestressed concrete bridge in Japan. They are depicted in

Figure 5.5. We can observe that these signals are 500 seconds long. As the sampling frequency is 1000 Hz, the number of actual data samples in each signal is $500K$. Note that the two accelerometers are placed at different positions on the bridge. Therefore, the events (vibrations induced by passing vehicles) on these signals are not synchronized[1].

Evaluation measures. The ground truth, including the number of vehicles and sequences of time points that true events occur in the signals, was obtained from a video recording. For a particular signal collected from a single accelerometer, we denote the sequence of true time points as $\mathcal{T} = [t_1, \ldots, t_k]$, where k is the number of vehicles. For any method that performs detection on this signal, we denote the sequence of detected time points[2] as $\widehat{\mathcal{T}} = [\hat{t}_1, \ldots, \hat{t}_{\hat{k}}]$, where k and $\hat{k}$ are not necessarily equal. True positives TP are true time points for which there is a detected one at least less than ϵ samples, *i.e.*

$$TP(\mathcal{T}, \widehat{\mathcal{T}}) = \{t_i \mid \exists j \in \{1, \ldots, \hat{k}\} \text{ s.t. } |t_i - \hat{t}_j| < \epsilon\}. \tag{5.2}$$

In this work, we set the margin $\epsilon = 5$. The precision $PREC$ and recall REC are then given by

$$PREC(\mathcal{T}, \widehat{\mathcal{T}}) = \frac{|TP(\mathcal{T}, \widehat{\mathcal{T}})|}{\hat{k}}, \tag{5.3}$$

$$REC(\mathcal{T}, \widehat{\mathcal{T}}) = \frac{|TP(\mathcal{T}, \widehat{\mathcal{T}})|}{k}. \tag{5.4}$$

The $F1$ score is the harmonic mean of the precision and recall

$$F1(\mathcal{T}, \widehat{\mathcal{T}}) = 2 \times \frac{PREC(\mathcal{T}, \widehat{\mathcal{T}}) \times REC(\mathcal{T}, \widehat{\mathcal{T}})}{PREC(\mathcal{T}, \widehat{\mathcal{T}}) + REC(\mathcal{T}, \widehat{\mathcal{T}})}. \tag{5.5}$$

The above three measures are well defined. Their best values are 1 and their worst values are 0.

Comparison. We compute the precision, recall and $F1$ score for the following detection results:

- *Oscillations_1*: sequence of oscillations detected by the current method on the first signal

[1] Vibrations correspond to one vehicle occur at different time points on the two signals.
[2] Peaks detected by the new framework or oscillations detected by the current method.

without refinement.

- *Oscillations_2*: sequence of oscillations detected by the current method on the second signal without refinement.
- *Oscillations_ref*: sequence of oscillations detected by the current method on the first signal with refinement.
- *Peaks*_1*: sequence of peaks detected by the proposed framework on the first signal without refinement.
- *Peaks*_2*: sequence of peaks detected by the proposed framework on the second signal without refinement.
- *Peaks*_ref*: sequence of peaks detected by the proposed framework on the first signal with refinement.

In addition, to further investigate the contribution of the model GSCAEs to the performance of the proposed framework, we perform ablation study by substituting it with two separate autoencoders, each of which process a particular signal. Therefore, we have additional results of this substitution as listed below:

- *Peaks_1*: sequence of peaks detected by the proposed framework, where GSCAEs is substituted with two separate autoencoders, on the first signal without refinement.
- *Peaks_2*: sequence of peaks detected by the proposed framework, where GSCAEs is substituted with two separate autoencoders, on the second signal without refinement.
- *Peaks_ref*: sequence of peaks detected by the proposed framework, where GSCAEs is substituted with two separate autoencoders, on the first signal with refinement.

Results. All the evaluation scores are shown in Table 5.1. We can observe that the detection results returned by the new framework have better qualities than those of the current method. Note that the proposed framework and the current method are different in the first two phases.

Detection results	$\|TP\|/\|FP\|$	$PREC$	REC	$F1$
Oscillations_1	42/16	0.72	0.89	0.79
Oscillations_2	41/21	0.66	0.87	0.75
Oscillations_ref	42/13	0.76	0.89	0.82
Peaks_1	44/08	0.84	0.93	0.88
Peaks_2	45/17	0.72	0.95	0.81
Peaks_ref	45/05	0.90	0.95	0.92
Peaks*_1	47/00	**1.00**	**1.00**	**1.00**
Peaks*_2	47/06	0.88	**1.00**	0.93
Peaks*_ref	47/00	**1.00**	**1.00**	**1.00**

Table 5.1: Evaluation scores on different detection results.

More specifically, the current method extracts oscillatory components from the raw signal in the first phase and then performs oscillation detection in the second phase. In contrast, the new framework feeds wavelet coefficient matrices of the raw signals into the GSCAEs model and then detects peaks from the reconstruction errors. Since autoencoders-based model can account for different types of noise [Xia et al., 2015, Zhou and Paffenroth, 2017, Kieu et al., 2018, Kieu et al., 2019] and peaks detection is less complicated than detecting oscillations, the proposed framework can achieve higher accuracy than the compared method.

We also see that substituting the GSCAEs model with two separate autoencoders considerably degrades performance of the new framework. This result again validates the importance of the GSCAEs model. Different from the two separate autoencoders that process the signals independently, GSCAEs jointly reconstructs the signals in the wavelet space from a shared hidden representation. This allows complementary information from one signal to contribute to the reconstruction of the other signal. Therefore, GSCAEs can output more discriminative reconstruction errors, facilitating the peaks detection in the subsequent phase.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this dissertation, we consider two problems arise in sequential analysis that involve in optimization over discrete variables with combinatorial constraints. The first problem is finding correspondence among samples while learning representation from multi-view sequential data. Because the objective of the sequential correspondence problem is not differentiable, existing models often perform alignment and representation learning in an alternative manner. This approach is prone to suboptimal solutions. In addition, these models are difficult to be extended, especially for those employing deep neural networks for learning the new representation. The second problem is segmentation of data sequences. The optimal algorithm to solve this problem is based on dynamic programming, which has quadratic time and memory complexities. Thus, it is difficult to optimally segment long data sequences. Although approximation algorithms for sequentially segmenting the sequences have been proposed, the quality of their solutions is not guaranteed.

To address the aforementioned issues, we propose novel models, whose objective functions are differentiable almost everywhere. The benefits of the differentiability are: *i)* All the variables are now updated in a unified manner during optimization. Thus, sub-optimal solutions are likely to be avoided. *ii)* Stochastic gradient descent is now applicable to train or learn the models'

parameters. This mitigates the computational time and memory requirement burdens. *iii)* The models are now more extendable to handle multiple data sequences. In summary, achievements of this dissertation are:

Differentiable models for sequential representation learning. First, we introduce *Generalized sequential correlation analysis* (GSCA) – a deep model for learning representation from multi-view sequential data. Our model can implicitly discover sample correspondences between data sequences while learning the new representation. Thanks to the differentiability of the objective of GSCA, the optimal correspondence and representation are obtained in an unified manner, avoiding sub-optimal solutions. Second, we next combine GSCA with reconstruction losses of autoencoders to form the second model termed *Generalized sequentially correlated autoencoders* (GSCAEs). GSCAEs offers a better trade-off between within-sequence and cross-sequence relations for sequential representation learning. Third, we finally develop an extension of GSCA termed *Generalized multiple sequences analysis* GMSA for simultaneously handling multiple data sequences.

Differentiable models for segmentation of sequences. First, we introduce *Kernel clustering with sigmoid regularization* (KCSR) – a segmentation model based on kernel clustering. KCSR is a combination of a novel regularization based on sigmoid functions and kernel $k-$means. The objective function of our model is differentiable almost everywhere. Thus, it can be effectively minimized using gradient descent algorithm. Second, to further improve the efficiency, we propose a stochastic variant of KCSR termed *Stochastic kernel clustering with sigmoid regularization* (SKCSR). This model employs stochastic gradient descent, where the gradient is estimated from a randomly sampled subsequence, for optimization. Thus, it has much lower time and space complexities than existing methods. SKCSR is especially free from storing large affinity and/or kernel matrices. Hence, it can handle large-scale sequential data. Third, we extend KCSR by slightly modifying the sigmoid-based regularization, forming the new model termed *Multiple kernel clustering with sigmoid regularization* (MKCSR). Inheriting good properties from KCSR, MKCSR can effectively and efficiently segment multiple data sequences simultaneously.

Performances of all the proposed models in this thesis are evaluated on various widely public datasets and compared with those of existing methods. Extensively experimental results validate the claimed advantages of our models. Finally, we discuss limitations of the proposed models and indicate several potential directions for extending the current research.

6.2 Future Work

In this section, we discuss open issues and directions to extend the current research.

Sequential representation learning. In chapter 3, we proposed three differentiable models for learning representation from multi-view sequential data. These models handle the misalignment in the data sequences mainly based on generalized smooth DTW – a differentiable approximation of the original DTW. However, due to replacing the hard minimum in DTW with a smooth one, generalized smooth DTW induces some entropic biases to the alignment [Blondel et al., 2021]. To improve performance of the proposed models, it is crucial to modify the generalized smooth DTW, mitigating negative influences of the biases on the representation learning.

Another direction for extending the current model is to take into account local structures of the data sequences to improve both alignment accuracy and representation goodness. More specifically, current models are employing DTW to achieve the sample correspondence among data sequences. However, DTW is a frame-to-frame matching algorithm. Thus, it ignores the locality among the sequential data points. In case the data sequences have a particular structure (*e.g.*, they can be divided into several homogeneous segments), this information should be exploited to improve efficiency and effectiveness of the alignment procedure. Furthermore, local information can help to guide the representation learning procedure. This is often the case when supervised information is absent and by preserving locality, the learned representation is expected to be more robust.

Segmentation of sequences. In chapter 4, we introduced differentiable models for segmentation of a single or multiple data sequences. Although, owning many good properties

inherited from the differentiability of the objectives, there are still several remaining limitations in the proposed models. First, KCSR and its stochastic variant assume that the number segments k is given in advance. Therefore, the proposed models can only work under offline-settings. To further improve the adaptability, it is necessary to extend the current models such that the number of segments can be estimated from the data. Second, MKCSR – an extension of the original KCSR for segmentation of multiple sequences – can only operate in a single space. More precise, the data sequences to be segmented are required to have the same dimension. However, as nowadays, more and more sequential data are collected from multiple perspectives and they often have different modalities (*e.g.*, audio and video) the dimension requirement is likely to be violated. Motivated from this limitation, the next step of the current research is to develop a model that, on one hand, can preserve the goodness of MKCSR, while, on the other hand, can handle multimodal sequential data.

Appendix A

Smooth Min Operator

The smooth min operator is defined as:

$$\min_\Omega(\boldsymbol{\eta}) \coloneqq \min_{\boldsymbol{\gamma}\in\Delta^k} \langle \boldsymbol{\gamma}, \boldsymbol{\eta}\rangle + \frac{1}{\beta}\Omega(\boldsymbol{\gamma}), \tag{A.1}$$

where the regularization term $\Omega(\boldsymbol{\gamma})$ must be a strictly convex function [Nesterov, 2005]. Two widely used functions are Shannon entropy and squared ℓ_2 norm.

Shannon entropy. If $\Omega(\boldsymbol{\gamma}) = \sum_{i=1}^k \gamma_i \ln \gamma_i$, we obtain

$$\min_\Omega(\boldsymbol{\eta}) = \min_{\boldsymbol{\gamma}\in\Delta^k} \sum_{i=1}^k \gamma_i\eta_i + \frac{1}{\beta}\sum_{i=1}^k \gamma_i \ln \gamma_i. \tag{A.2}$$

Because the objective is strictly convex, we can take its Lagrangian:

$$L = \sum_{i=1}^k \gamma_i\eta_i + \frac{1}{\beta}\sum_{i=1}^k \gamma_i \ln \gamma_i + \lambda_1\left(1 - \sum_{i=1}^k \gamma_i\right) + \lambda_2 \sum_{i=1}^k \gamma_i. \tag{A.3}$$

With KKT conditions $\frac{\partial L}{\partial \gamma_i} = 0$ and slackness $\lambda_2\gamma_i = 0$, we have

$$\gamma_i = \mathrm{e}^{\beta\lambda_1 - \beta\eta_i - 1} \quad \forall i = 1, ..., k. \tag{A.4}$$

Combining with the simplex constraint: $\sum_{i=1}^{k} \gamma_i = 1$, we obtain

$$e^{\beta\lambda_1} = \frac{e}{\sum_{i=1}^{k} e^{-\beta\eta_i}}. \tag{A.5}$$

Plugging this back into equation (A.4), we arrive at the minimum of (A.2)

$$\gamma_i = \frac{e^{-\beta\eta_i}}{\sum_{j=1}^{k} e^{-\beta\eta_j}}. \tag{A.6}$$

In summary, when using Shannon entropy as regularization, we have closed-form solutions of the smooth min operator and its gradient

$$\min_\Omega(\boldsymbol{\eta}) = -\frac{1}{\beta} \ln \sum_{i=1}^{k} e^{-\beta\eta_i}, \tag{A.7}$$

$$\nabla\min_\Omega(\boldsymbol{\eta}) = \frac{e^{-\beta\boldsymbol{\eta}}}{\sum_{j=1}^{k} e^{-\beta\eta_j}}. \tag{A.8}$$

Squared ℓ_2 norm. When $\Omega(\boldsymbol{\gamma}) = \frac{1}{2}\sum_{i=1}^{k} \gamma_i^2$, the smooth min becomes

$$\min_\Omega(\boldsymbol{\eta}) = \min_{\boldsymbol{\gamma}\in\Delta^k} \sum_{i=1}^{k} \gamma_i\eta_i + \frac{1}{2\beta}\sum_{i=1}^{k} \gamma_i^2. \tag{A.9}$$

It can be easily shown that the minimum $\boldsymbol{\gamma}^*$ (i.e. $\nabla\min_\Omega(\boldsymbol{\eta})$) of (A.9) is the projection of $-\beta\boldsymbol{\eta}$ onto the simplex Δ^k

$$\boldsymbol{\gamma}^* = \operatorname*{argmin}_{\boldsymbol{\gamma}\in\Delta^k} ||-\beta\boldsymbol{\eta} - \boldsymbol{\gamma}||_2^2, \tag{A.10}$$

which is likely to be sparse. The solution of (A.10) can be efficiently obtained using the algorithm proposed in [Brucker, 1984, Pardalos and Kovoor, 1990, Duchi et al., 2008] with a complexity of $O(k \ln k)$.

Appendix B

Generalized Smooth DTW with Entropy Regularization

Theorem B.1. *Let* $\mathbf{\Pi}$ *denotes the set of all warping paths*

$$\boldsymbol{\pi} = \langle (i_1, j_1), \ldots, (i_p, j_p) \rangle, \tag{B.1}$$

that satisfies three conditions: Boundary, Continuity, and Monotonicity as described in subsection 2.1.2, and $\{s(\boldsymbol{\pi}) = d_{i_1,j_1} + \cdots + d_{i_p,j_p} | \boldsymbol{\pi} \in \mathbf{\Pi}\}$ *be a set of cumulative sums corresponding to all the warping paths. If the regularization* Ω *is the Shannon entropy, then*

$$\begin{aligned} DTW_{\Omega}(\boldsymbol{X}, \boldsymbol{Y}) &= DTW_{\beta}(\boldsymbol{X}, \boldsymbol{Y}) \\ &= -\frac{1}{\beta} \ln \sum_{\boldsymbol{\pi} \in \Pi} \mathrm{e}^{-\beta s(\boldsymbol{\pi})} \,. \end{aligned} \tag{B.2}$$

Proof. Let $\mathbf{\Pi}_{i,j} \subset \mathbf{\Pi}$ be the set of all warping paths from $(1,1)$ to (i,j), and denotes

$$r_{i,j} = -\frac{1}{\beta} \ln \sum_{\boldsymbol{\pi}_0 \in \mathbf{\Pi}_{i,j}} \mathrm{e}^{-\beta s(\boldsymbol{\pi}_0)} \,. \tag{B.3}$$

Note that when the regularization Ω is the Shannon entropy smooth min has a closed-form

expression as shown in equation (A.7), thus we also have:

$$r_{i,j} = \min_{\Omega}\big(\{s(\boldsymbol{\pi}_0)|\boldsymbol{\pi}_0 \in \boldsymbol{\Pi}_{i,j}\}\big). \tag{B.4}$$

We can rewrite equation (B.3) as follow:

$$\begin{aligned} r_{i,j} &= -\frac{1}{\beta}\ln\left(\sum_{\boldsymbol{\pi}_1\in\boldsymbol{\Pi}_{i-1,j}} \mathrm{e}^{-\beta\left(s(\boldsymbol{\pi}_1)+d_{i,j}\right)} + \sum_{\boldsymbol{\pi}_2\in\boldsymbol{\Pi}_{i,j-1}} \mathrm{e}^{-\beta\left(s(\boldsymbol{\pi}_2)+d_{i,j}\right)} + \sum_{\boldsymbol{\pi}_3\in\boldsymbol{\Pi}_{i-1,j-1}} \mathrm{e}^{-\beta\left(s(\boldsymbol{\pi}_3)+d_{i,j}\right)}\right) \\ &= -\frac{1}{\beta}\ln \mathrm{e}^{-\beta d_{i,j}}\left(\sum_{\boldsymbol{\pi}_1\in\Pi_{i-1,j}} \mathrm{e}^{-\beta s(\boldsymbol{\pi}_1)} + \sum_{\boldsymbol{\pi}_2\in\boldsymbol{\Pi}_{i,j-1}} \mathrm{e}^{-\beta s(\boldsymbol{\pi}_2)} + \sum_{\boldsymbol{\pi}_3\in\boldsymbol{\Pi}_{i-1,j-1}} \mathrm{e}^{-\beta s(\boldsymbol{\pi}_3)}\right) \\ &= d_{i,j} + -\frac{1}{\beta}\ln\left(\sum_{\boldsymbol{\pi}_1\in\boldsymbol{\Pi}_{i-1,j}} \mathrm{e}^{-\beta s(\boldsymbol{\pi}_1)} + \sum_{\boldsymbol{\pi}_2\in\boldsymbol{\Pi}_{i,j-1}} \mathrm{e}^{-\beta s(\boldsymbol{\pi}_2)} + \sum_{\boldsymbol{\pi}_3\in\boldsymbol{\Pi}_{i-1,j-1}} \mathrm{e}^{-\beta s(\boldsymbol{\pi}_3)}\right). \end{aligned} \tag{B.5}$$

Using the expression in equation (A.7) again, we obtain

$$\sum_{\boldsymbol{\pi}_1\in\boldsymbol{\Pi}_{i-1,j}} \mathrm{e}^{-\beta s(\boldsymbol{\pi}_1)} = \mathrm{e}^{\ln\sum_{\boldsymbol{\pi}_1\in\boldsymbol{\Pi}_{i-1,j}} \mathrm{e}^{-\beta s(\boldsymbol{\pi}_1)}} \tag{B.6}$$

$$= \mathrm{e}^{-\beta \min_{\Omega}\left(\{s(\boldsymbol{\pi}_1)|\boldsymbol{\pi}_1\in\boldsymbol{\Pi}_{i-1,j}\}\right)} \tag{B.7}$$

$$= \mathrm{e}^{-\beta r_{i-1,j}}\,. \tag{B.8}$$

The similar expressions for the sums over $\boldsymbol{\pi}_2 \in \boldsymbol{\Pi}_{i,j-1}$ and $\boldsymbol{\pi}_3 \in \boldsymbol{\Pi}_{i-1,j-1}$ can be derived in the same manner. Substituting (B.8) into (B.5), we have

$$r_{i,j} = d_{i,j} + \min_{\Omega}(r_{i-1,j}, r_{i,j-1}, r_{i-1,j-1}). \tag{B.9}$$

By recursively applying equation (B.9) for $i = 1..., n$ and $j = 1, ..., m$, we can arrive at equation (B.2), completing the proof. □

Appendix C

Forward-backward Algorithm

Algorithm 5 : Forward-backward algorithm

Require: Distance matrix $\boldsymbol{D} \in \mathbb{R}^{n \times m}$

Ensure: Gradient matrix $\boldsymbol{E} = \frac{\partial \mathrm{DTW}_\Omega(\boldsymbol{X}, \boldsymbol{Y})}{\partial \boldsymbol{D}} \in \mathbb{R}^{n \times m}$.

▷ Forward pass:

$s'_{0,0} = 0, s'_{i,0} = s'_{0,j} = \infty \; \forall i, j.$

for $i = 1, \ldots, n$ and $j = 1, \ldots, m$ **do**

  $s'_{i,j} = d_{i,j} + \min_\Omega(s'_{i-1,j}, s'_{i,j-1}, s'_{i-1,j-1})$

  $\boldsymbol{q}_{i,j} = \nabla \min_\Omega(s'_{i-1,j}, s'_{i,j-1}, s'_{i-1,j-1}) \in \mathbb{R}^3$

end for

▷ Backward pass:

$\boldsymbol{q}_{i,m+1} = \boldsymbol{q}_{n+1,j} = \boldsymbol{0}_3, e_{i,m+1} = e_{n+1,j} = 0 \; \forall i, j.$

$\boldsymbol{q}_{n+1,m+1} = [0, 1, 0], e_{n+1,m+1} = 1.$

for $i = 1, \ldots, n$ and $j = 1, \ldots, m$ **do**

  $e_{i,j} = q_{i,j+1,1} e_{i,j+1} + q_{i+1,j+1,2} e_{i+1,j+1} + q_{i+1,j,3} e_{i+1,j}$

end for

To compute $e_{i,j}$ in equation (3.9), we use the forward-backward algorithm, which is originally introduced in [Mensch and Blondel, 2018]. The details are shown in Algorithm 5. The algorithm indeed computes the gradient matrix $\boldsymbol{E}$, where $e_{i,j}$ is the element at position (i, j), of the generalized smooth DTW with regard to the distance matrix $\boldsymbol{D}$. It includes a forward step and a backward step. Both of them perform constant-time operations in nm times. Therefore, the computational complexity of the algorithm is $O(nm)$. In addition, during the computation, the algorithm stores several matrices whose largest size is $3nm$. Thus, its space complexity is also $O(nm)$. Note that when the squared ℓ_2 norm is used as regularization in DTW_Ω, $\boldsymbol{q}_{i,j}$ become sparse because of equation (A.10). This then induces the sparsity in $\boldsymbol{E}$, further reducing

the complexity of the algorithm in terms of both time and space.

Appendix D

Update Rule for Consensus Label Sequence

In this section, we provide the derivation of the update rule for the consensus label sequence in equation (3.19). By adding an extra term $\xi||\boldsymbol{Z}\boldsymbol{Z}^\top - \boldsymbol{I}||_F^2$ and introducing a Lagrange multiplier matrix $\boldsymbol{\Psi} \in \mathbb{R}^{c\times n}$, we have the following Lagrange function

$$\mathcal{L}(\boldsymbol{Z}, \boldsymbol{\Psi}) = \sum_{k=1}^{v} \mathrm{DTW}_\Omega(\boldsymbol{Z}, \boldsymbol{Z}^{(k)}) + \xi||\boldsymbol{Z}\boldsymbol{Z}^\top - \boldsymbol{I}||_F^2 + \mathrm{Tr}(\boldsymbol{\psi}^\top \boldsymbol{Z}). \tag{D.1}$$

Taking the derivative of $\mathcal{L}(\boldsymbol{Z}, \boldsymbol{\psi})$ with regard to $\boldsymbol{Z}$ and setting it to zero, we obtain

$$\frac{\partial \mathcal{L}(\boldsymbol{Z}, \boldsymbol{\psi})}{\partial \boldsymbol{Z}} = \sum_{k=1}^{v} \frac{\partial \mathrm{DTW}_\Omega(\boldsymbol{Z}, \boldsymbol{Z}^{(k)})}{\partial \boldsymbol{Z}} + 4\xi(\boldsymbol{Z}\boldsymbol{Z}^\top - \boldsymbol{I})\boldsymbol{Z} + \boldsymbol{\Psi} = 0. \tag{D.2}$$

Then

$$\boldsymbol{\Psi} = 4\xi\boldsymbol{Z} - 4\xi\boldsymbol{Z}\boldsymbol{Z}^\top\boldsymbol{Z} - \boldsymbol{G}, \tag{D.3}$$

where $\boldsymbol{G} = \sum_{k=1}^{v} \frac{\partial \mathrm{DTW}_\Omega(\boldsymbol{Z}, \boldsymbol{Z}^{(k)})}{\partial \boldsymbol{Z}}$. According to the Karush—Kuhn–Tucker condition [Boyd et al., 2004], i.e. $\psi_{i,j} z_{i,j} = 0$, we can arrive at the following equation:

$$[4\xi\boldsymbol{Z} - 4\xi\boldsymbol{Z}\boldsymbol{Z}^\top\boldsymbol{Z} - \boldsymbol{G}]_{i,j} z_{i,j} = 0. \tag{D.4}$$

Then, we obtain the update rule for $\boldsymbol{Z}$:

$$z_{i,j} \leftarrow z_{i,j} \frac{[4\xi \boldsymbol{Z}]_{i,j}}{[\boldsymbol{G} + 4\xi \boldsymbol{Z}\boldsymbol{Z}^\top \boldsymbol{Z}]_{i,j}}. \tag{D.5}$$

Appendix E

Derivation of The Gradient for KCSR

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

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

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

$$\frac{\partial J_{KCSR}}{\partial \boldsymbol{G}} = 2\left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1} \boldsymbol{G}\boldsymbol{K}\boldsymbol{G}^\top \left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1} \boldsymbol{G} - 2\left(\boldsymbol{G}\boldsymbol{G}^\top\right)^{-1} \boldsymbol{G}\boldsymbol{K} + \lambda \boldsymbol{G}\mathbf{1}\mathbf{1}^\top. \tag{E.2}$$

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

$$\frac{\partial G_{j.i}}{\partial \tau_i} = \frac{\partial \max\left(0, 1 - |\tau_i - j|\right)}{\partial \tau_i} = \begin{cases} -1 & \text{if } j \leq \tau_i \leq j+1 \\ 1 & \text{if } j-1 \leq \tau_i < j \\ 0 & \text{otherwise.} \end{cases} \tag{E.3}$$

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

$$\frac{\partial J_{KCSR}}{\partial \tau_i} = \sum_{j=1}^{k} \frac{\partial J_{KCSR}}{\partial G_{j.i}} \frac{\partial G_{j.i}}{\partial \tau_i}. \tag{E.4}$$

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

$$\begin{aligned}\frac{\partial \tau_i}{\partial \beta_j} &= \frac{\partial \left(1 + \sum_{j'=1}^{k-1} \left(1 + \mathrm{e}^{-\alpha(i-\beta_{j'})}\right)^{-1}\right)}{\partial \beta_j} \\ &= -\alpha \left(1 + \mathrm{e}^{-\alpha(i-\beta_j)}\right)^{-1} \left[1 - \left(1 + \mathrm{e}^{-\alpha(i-\beta_j)}\right)^{-1}\right]. \end{aligned} \tag{E.5}$$

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

$$\frac{\partial J_{KCSR}}{\partial \beta_j} = \sum_{i=1}^{n} \frac{\partial J_{KCSR}}{\partial \tau_i} \frac{\partial \tau_i}{\partial \beta_j}. \tag{E.6}$$

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

$$\frac{\partial J_{KCSR}}{\partial \gamma_c} = \sum_{j=1}^{k-1} \frac{\partial J_{KCSR}}{\partial \beta_j} \frac{\partial \beta_j}{\partial \gamma_c}, \tag{E.7}$$

where

$$\frac{\partial \beta_j}{\partial \gamma_c} = \begin{cases} \frac{(n-1)\,\mathrm{e}^{\gamma_c}}{\sum_{j'=1}^{k} \mathrm{e}^{\gamma_{j'}}} \left(1 - \frac{\sum_{j'=1}^{j} \mathrm{e}^{\gamma_{j'}}}{\sum_{j'=1}^{k} \mathrm{e}^{\gamma_{j'}}}\right) & \text{if } c \leq j \\ -\frac{(n-1)\,\mathrm{e}^{\gamma_c} \sum_{j'=1}^{j} \mathrm{e}^{\gamma_{j'}}}{\left(\sum_{j'=1}^{k} \mathrm{e}^{\gamma_{j'}}\right)^2} & \text{if } c > j \end{cases}. \tag{E.8}$$

Appendix F

Definitions of Accuracy and Normalized Mutual Information

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

$$ACC = \frac{\sum_{i=1}^{n} \delta(l_i, map(\hat{l}_i))}{n}, \tag{F.1}$$

where $\delta(a, b)$ is the delta function that equals one if $a = b$ and zero otherwise and $map(\hat{l}_i)$ is the permutation mapping function that maps label $\hat{l}_i$ to the equivalent ground truth label. In this work, we use Kuhn-Munkres algorithm [Plummer, 1986] 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

$$MI(\mathcal{C}, \widehat{\mathcal{C}}) = \sum_{c_j \in \mathcal{C}, \hat{c}_{j'} \in \widehat{\mathcal{C}}} p(c_j, \hat{c}_{j'}) \log_2 \frac{p(c_j, \hat{c}_{j'})}{p(c_j) p(\hat{c}_{j'})}, \tag{F.2}$$

where $p(c_j)$ and $p(\hat{c}_{j'})$ are the probabilities that a data sample arbitrarily selected from the sequence belongs to the clusters c_j and $\hat{c}_{j'}$, respectively, and $p(c_j, \hat{c}_{j'})$ is the joint probability

that the selected data sample belongs to both c_j and $\hat{c}_{j'}$. This metric is normalized to the range $[0, 1]$ as follows:

$$NMI(\mathcal{C}, \widehat{\mathcal{C}}) = \frac{MI(\mathcal{C}, \widehat{\mathcal{C}})}{max(H(\mathcal{C}), H(\widehat{\mathcal{C}}))}, \tag{F.3}$$

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

Bibliography

[Abeo et al., 2019] Abeo, T. A., Shen, X.-J., Ganaa, E. D., Zhu, Q., Bao, B.-K., and Zha, Z.-J. (2019). Manifold alignment via global and local structures preserving pca framework. *IEEE Access*, 7:38123–38134.

[Aminikhanghahi and Cook, 2017] Aminikhanghahi, S. and Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367.

[Andrew et al., 2013] Andrew, G., Arora, R., Bilmes, J., and Livescu, K. (2013). Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255. PMLR.

[Arlot et al., 2019] Arlot, S., Celisse, A., and Harchaoui, Z. (2019). A kernel multiple change-point algorithm via model selection. *Journal of machine learning research*, 20(162).

[Armijo, 1966] Armijo, L. (1966). Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3.

[Bellman, 1961] Bellman, R. (1961). On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284.

[Bengio et al., 2013] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[Blondel et al., 2021] Blondel, M., Mensch, A., and Vert, J.-P. (2021). Differentiable divergences between time series. In *International Conference on Artificial Intelligence and Statistics*, pages 3853–3861. PMLR.

[Bottou, 1991] Bottou, L. (1991). Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12.

[Bottou, 1998] Bottou, L. (1998). Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142.

[Boyd et al., 2004] Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Brucker, 1984] Brucker, P. (1984). An o (n) algorithm for quadratic knapsack problems. *Operations Research Letters*, 3(3):163–166.

[Cai et al., 2005] Cai, D., He, X., and Han, J. (2005). Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637.

[Celisse et al., 2018] Celisse, A., Marot, G., Pierre-Jean, M., and Rigaill, G. (2018). New efficient algorithms for multiple change-point detection with reproducing kernels. *Computational Statistics & Data Analysis*, 128:200–220.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27.

[Chang et al., 2019] Chang, C.-Y., Huang, D.-A., Sui, Y., Fei-Fei, L., and Niebles, J. C. (2019). 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*, pages 3546–3555.

[Chen and Gupta, 2011] Chen, J. and Gupta, A. K. (2011). *Parametric statistical change point analysis: with applications to genetics, medicine, and finance*. Springer Science & Business Media.

[Clopton et al., 2017] Clopton, L., Mavroudi, E., Tsakiris, M., Ali, H., and Vidal, R. (2017). Temporal subspace clustering for unsupervised action segmentation. *CSMR REU*, pages 1–7.

[Cui et al., 2014] Cui, Z., Chang, H., Shan, S., and Chen, X. (2014). Generalized unsupervised manifold alignment. *Advances in Neural Information Processing Systems*, 27:2429–2437.

[Cuturi and Blondel, 2017] Cuturi, M. and Blondel, M. (2017). Soft-dtw: a differentiable loss function for time-series. In *International Conference on Machine Learning*, pages 894–903. PMLR.

[Danskin, 1966] Danskin, J. M. (1966). The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14(4):641–664.

[Davis and Mermelstein, 1980] Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366.

[De la Torre, 2012] De la Torre, F. (2012). A least-squares framework for component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1041–1055.

[Deng et al., 2019] Deng, Y., Xie, Y., Li, Y., Yang, M., Du, N., Fan, W., Lei, K., and Shen, Y. (2019). Multi-task learning with multi-view attention for answer selection and knowledge base question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6318–6325.

[Dhillon et al., 2004] Dhillon, I. S., Guan, Y., and Kulis, B. (2004). Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556.

[Doan and Takasu, 2019] Doan, P. T. and Takasu, A. (2019). Sparse regression-based multiple sequence alignment. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1372–1377. IEEE.

[Doan and Atsuhiro, 2019] Doan, T. and Atsuhiro, T. (2019). Deep multi-view learning from sequential data without correspondence. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[Doan and Takasu, 2017] Doan, T. and Takasu, A. (2017). Robust vehicle detection from noisy acceleration signal for bridge monitoring systems. In *Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services*, pages 136–140.

[Du et al., 2006] Du, P., Kibbe, W. A., and Lin, S. M. (2006). Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, 22(17):2059–2065.

[Duchi et al., 2008] Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279.

[Eddy, 1998] Eddy, S. R. (1998). Profile hidden markov models. *Bioinformatics (Oxford, England)*, 14(9):755–763.

[Elad et al., 2005] Elad, M., Starck, J.-L., Querre, P., and Donoho, D. L. (2005). Simultaneous cartoon and texture image inpainting using morphological component analysis (mca). *Applied and Computational Harmonic Analysis*, 19(3):340–358.

[Fadili and Starck, 2005] Fadili, J. M. and Starck, J.-L. (2005). Sparse representations and bayesian image inpainting. In *Proceedings of International Conferences SPARS'05*, pages 4–pp.

[Fan et al., 2016] Fan, K., Mian, A., Liu, W., and Li, L. (2016). Unsupervised manifold alignment using soft-assign technique. *Machine Vision and Applications*, 27(6):929–942.

[Fearnhead, 2006] Fearnhead, P. (2006). Exact and efficient bayesian inference for multiple changepoint problems. *Statistics and computing*, 16(2):203–213.

[Gong and Medioni, 2011] Gong, D. and Medioni, G. (2011). Dynamic manifold warping for view invariant action recognition. In *2011 International Conference on Computer Vision*, pages 571–578. IEEE.

[Gorelick et al., 2007] Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. (2007). Actions as space-time shapes. *IEEE transactions on pattern analysis and machine intelligence*, 29(12):2247–2253.

[Gorelick et al., 2006] Gorelick, L., Galun, M., Sharon, E., Basri, R., and Brandt, A. (2006). Shape representation and classification using the poisson equation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1991–2005.

[Gretton et al., 2012] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

[Harchaoui and Cappé, 2007] Harchaoui, Z. and Cappé, O. (2007). Retrospective mutiple change-point estimation with kernels. In *2007 IEEE/SP 14th Workshop on Statistical Signal Processing*, pages 768–772. IEEE.

[Harchaoui et al., 2009a] Harchaoui, Z., Moulines, E., and Bach, F. R. (2009a). Kernel change-point analysis. In *Advances in neural information processing systems*, pages 609–616.

[Harchaoui et al., 2009b] Harchaoui, Z., Vallet, F., Lung-Yut-Fong, A., and Cappé, O. (2009b). A regularized kernel-based approach to unsupervised audio segmentation. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1665–1668. IEEE.

[Hardoon et al., 2004] Hardoon, D. R., Szedmak, S., and Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664.

[Hasan, 2009] Hasan, M. A. (2009). On multi-set canonical correlation analysis. In *2009 International Joint Conference on Neural Networks*, pages 1128–1133. IEEE.

[Hoai and De la Torre, 2012] Hoai, M. and De la Torre, F. (2012). Maximum margin temporal clustering. In *Artificial Intelligence and Statistics*, pages 520–528. PMLR.

[Hoai and De la Torre, 2014] Hoai, M. and De la Torre, F. (2014). Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hoffman et al., 2010] Hoffman, M., Bach, F. R., and Blei, D. M. (2010). Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864. Citeseer.

[Hong et al., 2019] Hong, D., Yokoya, N., Ge, N., Chanussot, J., and Zhu, X. X. (2019). Learnable manifold alignment (lema): A semi-supervised cross-modality learning framework for land cover and land use classification. *ISPRS journal of photogrammetry and remote sensing*, 147:193–205.

[Hotelling, 1936] Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

[Hu et al., 2020] Hu, W., Li, S., Zheng, W., Lu, Y., and Yu, G. (2020). Robust sequential subspace clustering via ℓ_1-norm temporal graph. *Neurocomputing*, 383:380–395.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

[Jia et al., 2016] Jia, C., Shao, M., and Fu, Y. (2016). Sparse canonical temporal alignment with deep tensor decomposition for action recognition. *IEEE Transactions on Image Processing*, 26(2):738–750.

[Kawano et al., 2019] Kawano, K., Koide, S., and Kutsuna, T. (2019). Canonical soft time warping. In *Asian Conference on Machine Learning*, pages 551–566. PMLR.

[Kieu et al., 2019] Kieu, T., Yang, B., Guo, C., and Jensen, C. S. (2019). Outlier detection for time series with recurrent autoencoder ensembles. In *IJCAI*, pages 2725–2732.

[Kieu et al., 2018] Kieu, T., Yang, B., and Jensen, C. S. (2018). Outlier detection for multidimensional time series using deep neural networks. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, pages 125–134. IEEE.

[King, 2009] King, D. E. (2009). Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758.

[Lavielle and Teyssiere, 2007] Lavielle, M. and Teyssiere, G. (2007). Adaptive detection of multiple change-points in asset price volatility. In *Long memory in economics*, pages 129–156. Springer.

[Le et al., 2018] Le, H., Tran, T., and Venkatesh, S. (2018). Dual memory neural computer for asynchronous two-view sequential learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1637–1645.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Lévy-Leduc et al., 2009] Lévy-Leduc, C., Roueff, F., et al. (2009). Detection and localization of change-points in high-dimensional network traffic data. *Annals of Applied Statistics*, 3(2):637–662.

[Li and Todorovic, 2020] Li, J. and Todorovic, S. (2020). Set-constrained viterbi for set-supervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10820–10829.

[Li et al., 2015a] Li, S., Li, K., and Fu, Y. (2015a). Temporal subspace clustering for human motion segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4453–4461.

[Li et al., 2015b] Li, S., Xie, Y., Dai, H., and Song, L. (2015b). M-statistic for kernel change-point detection. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[Li et al., 2019] Li, S., Xie, Y., Dai, H., and Song, L. (2019). Scan b-statistic for kernel change-point detection. *Sequential Analysis*, 38(4):503–544.

[Li et al., 2020] Li, W., Xue, J., Chen, Y., Zhang, X., Tang, C., Zhang, Q., and Gao, Y. (2020). Fuzzy granule manifold alignment preserving local topology. *IEEE Access*, 8:178695–178705.

[Li et al., 2018] Li, Y., Yang, M., and Zhang, Z. (2018). A survey of multi-view representation learning. *IEEE transactions on knowledge and data engineering*, 31(10):1863–1883.

[Liu et al., 2017a] Liu, H., Cheng, J., and Wang, F. (2017a). Sequential subspace clustering via temporal smoothness for sequential data segmentation. *IEEE Transactions on Image Processing*, 27(2):866–878.

[Liu et al., 2017b] Liu, H., Han, J., Nie, F., and Li, X. (2017b). Balanced clustering with least square regression. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2231–2237.

[Lung-Yut-Fong et al., 2012] Lung-Yut-Fong, A., Lévy-Leduc, C., and Cappé, O. (2012). Distributed detection/localization of change-points in high-dimensional network traffic data. *Statistics and Computing*, 22(2):485–496.

[Maidstone et al., 2017] Maidstone, R., Hocking, T., Rigaill, G., and Fearnhead, P. (2017). On optimal multiple changepoint algorithms for large data. *Statistics and computing*, 27(2):519–533.

[Maurer et al., 2003] Maurer, C. R., Qi, R., and Raghavan, V. (2003). A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270.

[Mensch and Blondel, 2018] Mensch, A. and Blondel, M. (2018). Differentiable dynamic programming for structured prediction and attention. In *International Conference on Machine Learning*, pages 3462–3471. PMLR.

[Nesterov, 2005] Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152.

[Ng et al., 2001] Ng, A., Jordan, M., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14:849–856.

[Ngiam et al., 2011] Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. (2011). Multimodal deep learning. In *ICML*.

[Nie et al., 2011] Nie, F., Zeng, Z., Tsang, I. W., Xu, D., and Zhang, C. (2011). Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11):1796–1808.

[Nie et al., 2016] Nie, L., Wang, Y., Zhang, X., Huang, X., and Luo, Z. (2016). Enhancing temporal alignment with autoencoder regularization. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4873–4879. IEEE.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

[Panagakis et al., 2013] Panagakis, Y., Nicolaou, M. A., Zafeiriou, S., and Pantic, M. (2013). Robust canonical time warping for the alignment of grossly corrupted sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 540–547.

[Panagakis et al., 2015] Panagakis, Y., Nicolaou, M. A., Zafeiriou, S., and Pantic, M. (2015). Robust correlated and individual component analysis. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1665–1678.

[Pantic et al., 2005] Pantic, M., Valstar, M., Rademaker, R., and Maat, L. (2005). Web-based database for facial expression analysis. In *2005 IEEE international conference on multimedia and Expo*, pages 5–pp. IEEE.

[Pardalos and Kovoor, 1990] Pardalos, P. M. and Kovoor, N. (1990). An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46(1):321–328.

[Plummer, 1986] Plummer, M. (1986). Matching theory. *Ann. Discrete Math*, 29.

[Polyak, 1987] Polyak, B. T. (1987). Introduction to optimization. optimization software. *Inc., Publications Division, New York*, 1.

[Qiu et al., 2019] Qiu, J., Wang, X., Fua, P., and Tao, D. (2019). Matching seqlets: An unsupervised approach for locality preserving sequence matching. *IEEE transactions on pattern analysis and machine intelligence*.

[Rabiner, 1993] Rabiner, L. (1993). Fundamentals of speech recognition. *Fundamentals of speech recognition*.

[Rajagopalan et al., 2016] Rajagopalan, S. S., Morency, L.-P., Baltrusaitis, T., and Goecke, R. (2016). Extending long short-term memory for multi-view structured learning. In *European Conference on Computer Vision*, pages 338–353. Springer.

[Rasti et al., 2020] Rasti, B., Hong, D., Hang, R., Ghamisi, P., Kang, X., Chanussot, J., and Benediktsson, J. A. (2020). Feature extraction for hyperspectral imagery: The evolution from shallow to deep: Overview and toolbox. *IEEE Geoscience and Remote Sensing Magazine*, 8(4):60–88.

[Reeves et al., 2007] Reeves, J., Chen, J., Wang, X. L., Lund, R., and Lu, Q. Q. (2007). A review and comparison of changepoint detection techniques for climate data. *Journal of applied meteorology and climatology*, 46(6):900–915.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[Seichepine et al., 2014] Seichepine, N., Essid, S., Févotte, C., and Cappé, O. (2014). Piecewise constant nonnegative matrix factorization. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6721–6725. IEEE.

[Selesnick, 2011a] Selesnick, I. W. (2011a). Sparse signal representations using the tunable q-factor wavelet transform. In *Wavelets and Sparsity XIV*, volume 8138, page 81381U. International Society for Optics and Photonics.

[Selesnick, 2011b] Selesnick, I. W. (2011b). Wavelet transform with tunable q-factor. *IEEE transactions on signal processing*, 59(8):3560–3575.

[Sha et al., 2018] Sha, L., Zhang, X., Qian, F., Chang, B., and Sui, Z. (2018). A multi-view fusion neural network for answer selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.

[Shimodaira et al., 2001] Shimodaira, H., Noma, K.-i., Nakai, M., and Sagayama, S. (2001). Dynamic time-alignment kernel in support vector machine. *Advances in neural information processing systems*, 14:921–928.

[Si and Yin, 2013] Si, Y.-W. and Yin, J. (2013). Obst-based segmentation approach to financial time series. *Engineering Applications of Artificial Intelligence*, 26(10):2581–2596.

[Spall, 2005] Spall, J. C. (2005). *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons.

[Starck et al., 2005a] Starck, J.-L., Elad, M., and Donoho, D. L. (2005a). Image decomposition via the combination of sparse representations and a variational approach. *IEEE transactions on image processing*, 14(10):1570–1582.

[Starck et al., 2005b] Starck, J.-L., Moudden, Y., Bobin, J., Elad, M., and Donoho, D. (2005b). Morphological component analysis. In *Wavelets XI*, volume 5914, page 59140Q. International Society for Optics and Photonics.

[Sun, 2013] Sun, S. (2013). A survey of multi-view machine learning. *Neural computing and applications*, 23(7):2031–2038.

[Terzi and Tsaparas, 2006] Terzi, E. and Tsaparas, P. (2006). Efficient algorithms for sequence segmentation. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 316–327. SIAM.

[Tierney et al., 2014] Tierney, S., Gao, J., and Guo, Y. (2014). Subspace clustering for sequential data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1019–1026.

[Trigeorgis et al., 2018] Trigeorgis, G., Nicolaou, M. A., Schuller, B. W., and Zafeiriou, S. (2018). Deep canonical time warping for simultaneous alignment and representation learning of sequences. *IEEE Computer Architecture Letters*, 40(05):1128–1138.

[Truong et al., 2019] Truong, C., Oudre, L., and Vayatis, N. (2019). Greedy kernel change-point detection. *IEEE Transactions on Signal Processing*, 67(24):6204–6214.

[Truong et al., 2020] Truong, C., Oudre, L., and Vayatis, N. (2020). Selective review of offline change point detection methods. *Signal Processing*, 167:107299.

[Tsypkin, 1973] Tsypkin, Y. Z. (1973). Foundations of the theory of learning systems.

[Tsypkin and Nikolic, 1971] Tsypkin, Y. Z. and Nikolic, Z. J. (1971). *Adaptation and learning in automatic systems*, volume 73. Academic Press New York.

[Tuia et al., 2014] Tuia, D., Volpi, M., and Camps-Valls, G. (2014). Unsupervised alignment of image manifolds with centrality measures. In *2014 22nd International Conference on Pattern Recognition*, pages 912–917. IEEE.

[Van der Maaten and Hinton, 2008] Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

[Vapnik, 1982] Vapnik, V. (1982). Estimation of dependences based on empirical data: Springer series in statistics (springer series in statistics).

[Verbesselt et al., 2010] Verbesselt, J., Hyndman, R., Newnham, G., and Culvenor, D. (2010). Detecting trend and seasonal changes in satellite image time series. *Remote sensing of Environment*, 114(1):106–115.

[Vert and Bleakley, 2010] Vert, J.-P. and Bleakley, K. (2010). Fast detection of multiple change-points shared by many signals using group lars. *Advances in neural information processing systems*, 23:2343–2351.

[Vu et al., 2012] Vu, H., Carey, C., and Mahadevan, S. (2012). Manifold warping: Manifold alignment over time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26.

[Wang and Mahadevan, 2008] Wang, C. and Mahadevan, S. (2008). Manifold alignment using procrustes analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1120–1127.

[Wang and Mahadevan, 2009] Wang, C. and Mahadevan, S. (2009). Manifold alignment without correspondence. In *IJCAI*, volume 2, page 3. Citeseer.

[Wang and Mahadevan, 2013] Wang, C. and Mahadevan, S. (2013). Manifold alignment preserving global geometry. In *IJCAI*, pages 1743–1749. Citeseer.

[Wang et al., 2017] Wang, H., Yang, W., Yuan, C., Ling, H., and Hu, W. (2017). Human activity prediction using temporally-weighted generalized time warping. *Neurocomputing*, 225:139–147.

[Wang et al., 2015a] Wang, W., Arora, R., Livescu, K., and Bilmes, J. (2015a). On deep multi-view representation learning. In *International conference on machine learning*, pages 1083–1092. PMLR.

[Wang et al., 2015b] Wang, W., Arora, R., Livescu, K., and Srebro, N. (2015b). Stochastic optimization for deep cca via nonlinear orthogonal iterations. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 688–695. IEEE.

[Warden, 2018] Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

[Werbos, 1990] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[Westbury et al., 1994] Westbury, J. R., Turner, G., and Dembowski, J. (1994). X-ray microbeam speech production database user's handbook. *University of Wisconsin*.

[Wu et al., 2015] Wu, F., Hu, Y., Gao, J., Sun, Y., and Yin, B. (2015). Ordered subspace clustering with block-diagonal priors. *IEEE transactions on cybernetics*, 46(12):3209–3219.

[Xia et al., 2015] Xia, Y., Cao, X., Wen, F., Hua, G., and Sun, J. (2015). Learning discriminative reconstructions for unsupervised outlier removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1519.

[Ye et al., 2007] Ye, J., Zhao, Z., and Wu, M. (2007). Discriminative k-means for clustering. *Advances in neural information processing systems*, 20:1649–1656.

[Zadeh et al., 2018] Zadeh, A., Liang, P. P., Mazumder, N., Poria, S., Cambria, E., and Morency, L.-P. (2018). Memory fusion network for multi-view sequential learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Zafeiriou et al., 2016] Zafeiriou, L., Antonakos, E., Zafeiriou, S., and Pantic, M. (2016). Joint unsupervised deformable spatio-temporal alignment of sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3382–3390.

[Zass and Shashua, 2005] Zass, R. and Shashua, A. (2005). A unifying approach to hard and probabilistic clustering. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 294–301. IEEE.

[Zhao et al., 2017] Zhao, J., Xie, X., Xu, X., and Sun, S. (2017). Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54.

[Zheng et al., 2021] Zheng, J., Yang, P., Shen, G., Chen, S., and Zhang, W. (2021). Enhanced low-rank constraint for temporal subspace clustering and its acceleration scheme. *Pattern Recognition*, 111:107678.

[Zhong and Ghosh, 2003] Zhong, S. and Ghosh, J. (2003). Model-based clustering with soft balancing. In *ICDM'03: Proceedings of the Third IEEE International Conference on Data Mining*, page 459.

[Zhou and Paffenroth, 2017] Zhou, C. and Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674.

[Zhou and De la Torre, 2015] Zhou, F. and De la Torre, F. (2015). Generalized canonical time warping. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):279–294.

[Zhou et al., 2012] Zhou, F., De la Torre, F., and Hodgins, J. K. (2012). Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):582–596.

[Zhou and Torre, 2009] Zhou, F. and Torre, F. (2009). Canonical time warping for alignment of human behavior. *Advances in neural information processing systems*, 22:2286–2294.