Contributions to Collective Dynamical Clustering-Modeling of Discrete Time Series

by

Chiying Wang

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Computer Science

by

April 28, 2016

APPROVED:

Professor Carolina Ruiz
Worcester Polytechnic Institute
Advisor

Professor Sergio A. Alvarez
Boston College
Co-Advisor

Professor Sonia Chernova
Worcester Polytechnic Institute
Committee Member

Professor Emmanuel O. Agu
Worcester Polytechnic Institute
Committee Member

Professor Craig Wills
Worcester Polytechnic Institute
Head of Department

Abstract

The analysis of sequential data is important in business, science, and engineering, for tasks such as signal processing, user behavior mining, and commercial transactions analysis. In this dissertation, we build upon the Collective Dynamical Modeling and Clustering (CDMC) framework for discrete time series modeling, by making contributions to clustering initialization, dynamical modeling, and scaling.

We first propose a modified Dynamic Time Warping (DTW) approach for clustering initialization within CDMC. The proposed approach provides DTW metrics that penalize deviations of the warping path from the path of constant slope. This reduces over-warping, while retaining the efficiency advantages of global constraint approaches, and without relying on domain dependent constraints.

Second, we investigate the use of semi-Markov chains as dynamical models of temporal sequences in which state changes occur infrequently. Semi-Markov chains allow explicitly specifying the distribution of state visit durations. This makes them superior to traditional Markov chains, which implicitly assume an exponential state duration distribution.

Third, we consider convergence properties of the CDMC framework. We establish convergence by viewing CDMC from an Expectation Maximization (EM) perspective. We investigate the effect on the time to convergence of our efficient DTW-based initialization technique and selected dynamical models. We also explore the convergence implications of various stopping criteria.

Fourth, we consider scaling up CDMC to process big data, using Storm, an open source distributed real-time computation system that supports batch and distributed data processing.

We performed experimental evaluation on human sleep data and on user web navigation data. Our results demonstrate the superiority of the strategies introduced in this dissertation over state-of-the-art techniques in terms of modeling quality and efficiency.

Acknowledgements

The growth of my knowledge over the last six years culminating with my dissertation is to a huge part due to the inspiration and guidance I received from my advisor, Professor Carolina Ruiz. She gave me the freedom to explore any topic in data mining research, provided sound directions at every turn, and the prompt feedback that pushed my research envelope. I have been fortunate to have her as my advisor. I express my sincere thanks for her support, advice, patience, and encouragement throughout my graduate studies. Her excellence in teaching as well as tackling complex research problems will always be my inspiration to continue my research in future.

I am hugely appreciative to my co-advisor, Professor Sergio A. Alvarez. He has been very helpful in fully advising me throughout Ph.D program milestones. His enthusiasm and comprehensive knowledge on the academic field inspire me to work through research challenges. I am also very thankful for his patience in setting research directions and instructing my writings in this dissertation.

I sincerely thank Professor Emmanuel O. Agu, and Professor Sonia Chernova for providing me valuable feedback during the various milestones in my Ph.D. Their insight and critique helped me improve the contents of this dissertation. I would like to thank Professor Neil T. Heffernan for his guidance during my research qualification. I am thankful for the financial support I have received from the computer science department.

I would like to thank AIRG team members, in particular Hao Wan and Amhedul Kabir for their insightful discussions. In addition, I would also like to thank Chuan Lei, Lei Cao, and Kaiyu Zhao for their help in my Ph.D. study. I very much appreciated the discussions as well as friendship of Xiao Qin, Yanqing Zhou, Zhongfang Zhuang, Dongqing Xiao, and Hongzhu Cui.

I would like to thank Worcester Polytechnic Institute. It gave me the chance of embarking on a wonderful adventure in my quest for academic and intellectual challenges through postgraduate education. I gain a strengthened interest in establishing a industrial career in undertaking advanced research and development work at the forefront of machine learning, a dynamic and rapidly expanding field.

I would like to thank my parents, Kai Wang and Shufen Zhang, for their patience and support during the past few years. Their confidence in me and encouragement was in the end what made this dissertation possible. Their passion to achieve bigger and better things ingrained in me a drive to reach excellence.

Contents

List of Figures **vi**

List of Tables **viii**

1 Introduction **1**

1.1 Background 1

1.2 Motivation 4

1.3 Research Challenges Addressed in This Dissertation 7

1.4 Proposed Solutions 11

1.5 Dissertation Organization 13

2 Clustering of Time Series **14**

2.1 Dynamic Time Warping (DTW) 16

2.2 Dynamic Time Warping Clustering (DTWC) 20

2.3 Deviated Dynamic Time Warping 21

2.3.1 Deviation Measure 24

2.3.2 Deviation-Based Dynamic Time Warpings 26

2.4 Experimental Evaluation 31

2.4.1 Experimental Setup & Methodology 31

2.4.2 DTW-Driven Clustering Evaluation 35
2.4.3 Deviation-Based DTW Clustering Evaluation 35

3 Dynamical Modeling over Time Series 40

3.1 Markov Models 41
3.1.1 Markov Chains 42
3.2 Semi-Markov Models 43
3.2.1 State Bout Duration Distribution 44
3.2.2 Semi-Markov Chains With Explicit Bout Duration 49
3.3 Experimental Evaluation 51
3.3.1 Experimental Setup & Methodology 51
3.3.2 Markov Chain Evaluation 56
3.3.3 Semi-Markov Chain Evaluation 57

4 Convergence of the CDMC Framework 62

4.1 Expectation-Maximization Algorithm 63
4.2 Convergence Properties of CDMC Framework 65
4.3 Stopping Criteria 70
4.3.1 Rand Index 71
4.3.2 Adjusted Rand Index 71
4.3.3 Normalized Mutual Information 72
4.4 Experimental Evaluation 73
4.4.1 Experimental Setup & Methodology 74
4.4.2 Convergence For Stopping Criteria 75

5 Distributed Deployment Architecture 78

5.1 Apache Storm 80

5.2 Distributed Deployment on Storm 82
5.2.1 Distributed CDMC System Design 83
5.2.2 Distributed CDMC System Implementation 85
5.3 Experimental Evaluation 96
5.3.1 Experimental Setup & Methodology 96
5.3.2 Time Performance 98

6 Applications 101

6.1 Human Sleep Patterns 101
6.1.1 Data Description 103
6.1.2 Dynamical Model Evaluation 104
6.1.3 CDMC Framework Evaluation 106
6.2 User Navigation Behavior 111
6.2.1 Data Description 112
6.2.2 Dynamical Model Evaluation 114
6.2.3 CDMC Framework Evaluation 116

7 Conclusions of This Dissertation 123

8 Future Work 126

References 128

List of Figures

1.1 Sample Hypnogram . . . 3

1.2 Speech Recognition . . . 5

1.3 Variances of Models On Sequences With Infrequent Dynamic Events . . . 7

1.4 Efficient & Scalable CDMC Framework . . . 11

2.1 Four Time Series Clustering Approaches . . . 15

2.2 Variants of DTW . . . 19

2.3 DTWs Between Patients . . . 22

2.4 Deviation Measure . . . 25

2.5 DTW And DTW with SQRT-Deviation . . . 27

2.6 Accuracies for Variants of CDMC & DTWC . . . 36

2.7 DTW And DTW with Deviations . . . 38

2.8 Negative Log-Likelihoods of Dynamical Models . . . 39

3.1 Markov Chain With Exponential Decay . . . 43

3.2 State Bout Duration Distributions . . . 45

3.3 Weibull Distributions . . . 47

3.4 Kernel Density Estimation For Sleep Dataset . . . 48

3.5 A 3-State Semi-Markov Chain Model (SMM) . . . 50

3.6 Curve Fitting of Sleep Stage Duration Distribution In WNR 52
3.7 Curve Fitting of Sleep Stage Duration Distribution In WDL 53
3.8 Curve Fitting of Sleep Stage Duration Distribution In W5 54
3.9 Curve Fitting of The REM Stage Duration Distribution 61

4.1 Local & Global Extremes Across Parameter Space 73
4.2 Iterations To Convergence for Different Stopping Criteria 75
4.3 Accuracies for Different Stopping Criteria of Variants of CDMC 76

5.1 An Example of A Topology 82
5.2 Distributed Collective Dynamical Modeling & Clustering Framework Design 84
5.3 Time Performances of Two Applications in Distributed CDMC 99
5.4 Distributed Collective Dynamical Modeling & Clustering Framework Implementation 100

6.1 Introduction to Human Sleep Dataset 104
6.2 Comparison of An Original Dataset Hypnogram & Generations 105
6.3 Visualization of CDMC Clustering of Human Sleep Data 107
6.4 Negative Log-Likelihoods of Dynamical Models over Sleep Patterns . . . 108
6.5 Visualization of Human Sleep 110
6.6 The Cumulative Distribution of Session Lengths Over All Users 114
6.7 Distribution of Page Browsing History 115
6.8 Exponential Plots Over All Categories of Short-Term Visiting History . . 119
6.9 Weibull Plots Over All Categories of Short-Term Visiting History 120
6.10 Exponential Plots Over All Categories of Long-Term Visiting History . . 121
6.11 Weibull Plots Over All Categories of Long-Term Visiting History 122

List of Tables

1.1 A Sample of User Behavior Sequences 4

2.1 T-Test of Accuracies Between CDMC-rnd & DTWC 35

2.2 Median Accuracies of Clusterings Based on DTW, gwDTW, and sdDTW 38

3.1 Stage Transition Matrix For Markov Chain Model 56

3.2 Equilibrium Distribution For Markov Chain Model 56

3.3 Stage Transition Matrix For Semi-Markov Chain Model 57

3.4 Goodness of Fit (GOF) Values 59

3.5 Parameter Values for The Weibull Function Fits 59

3.6 Equilibrium Distribution For Semi-Markov Chain Model 60

5.1 Distributed Collective Dynamical Modeling Clustering System 85

5.2 A Summary of Caches in Distributed CDMC System 94

6.1 T-Test of Coordinates on Human Sleep Dataset 106

6.2 T-Tests of Generative Log-Likelihoods of Hypnograms on Dynamical Models 109

6.3 Median Negative Log Likelihoods of DTW Clusterings over Human Sleep Datasets 111

6.4 Encodings of User Navigation Behavior Categories 112
6.5 Statistics of MSNBC Dataset 113
6.6 T-Tests of Generative Log Likelihoods of Web User Navigation Behavior Dataset on Dynamical Models. 117
6.7 Median Negative Log Likelihoods of DTW Clusterings over Web User Navigation Behavior Datasets 118

Chapter 1

Introduction

1.1 Background

Time series constitute an essential data representation for images, video, human voice, and object motion, among others [1]. The analysis of data over time [2] thus has increasingly attracted attentions both from academia and industry. Among different methods of exploring time series, modeling symbolic representations of time series is particularly important [3]. The reasons behind it come down to four facets. First, symbolic data sequences are inherently well-defined data structures to represent the input to data analysis techniques such as neural networks [4]. Second, many effective and widely investigated data mining algorithms are specially designed for discrete time series [5]. Third, numerous techniques for transforming real-valued time series into discrete representations have been examined over the past decades [6]. Fourth, extensive experimentation with discrete time series has shown a wealth of successful applications in clustering, classification, anomaly detection, and other areas [7]. Without doubt, the investigation of discrete time series is a topic of great importance in the data mining community.

In addition to theoretical work, a wide range of practical applications with respect to discrete temporal data has been proposed. They span signal processing, biological sequence analysis, user behavior mining, human speech recognition, weather forecasting, and commercial transactions analysis for example. Among them, one important application is human sleep data analysis. As illustrated in Figure 1.1, the dynamics of human sleep [8], including sleep stage transition and sleep stage duration, have been essential indicators for describing the relationship between human sleep and human health [9, 10, 11]. The discrete sleep stages can be modeled as states of a dynamic process, with each stage as one state. Furthermore, the Markov property and time-homogeneous nature of the transition probability distribution hold at least approximately, although actual empirical data shows some non-stationary properties [12, 13, 14]. Based on those observations, Markov chains have been used to model the dynamics of sleep stage transitions. A simple time-homogeneous Markov chain was first applied in the sleep domain [14]. However, Markov chains (and more general, hidden Markov models) do not model sleep stage transitions accurately, because these models force geometrically distributed stage bout durations for all sleep stages, contradicting known experimental observations [10, 15]. Other state-duration based models are provided as alternatives of describing human sleep [16].

Another popular application regarding symbolic data representation is user behavior pattern mining. Table 1.1 shows a sample of user behavior sequences [20]. The web server logs for a certain period typically produces a collection of such sequences. Increasing web viewing history and online data transmission have resulted in a great effort on web prediction modeling. The prediction mining process aims at predicting online user requests ahead of time and creating a robust web information service. These anticipated requests are addressed by either storing pages at the server side or sending the response to the client in advance to reduce web latency. To model user viewing history, each viewed webpage is thought of as a state and each switch from one page to another page is a state transition. Under such transformations, state-based models (e.g., Markov-based predic-

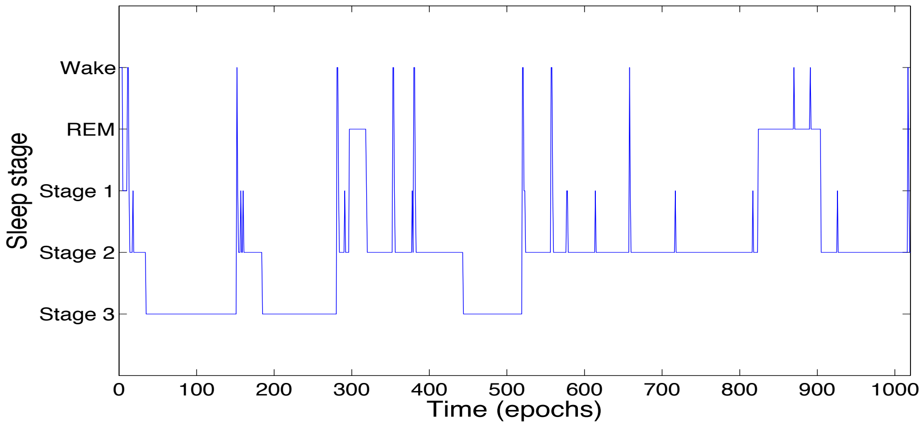


Figure 1.1: A sample diagram of the distribution and time evolution of sleep stages. It consists of 1,020 30-second epochs, which amount to 8.5 hours of sleep. Sleep progression generally starts with the wake stage [17] and then there are cycles in which REM [18] and NREM (stages 1, 2, and 3) alternate, particularly between REM stage and stage 2. During the whole night sleep recording, stage 2 exhibits long uninterrupted bouts as the night goes on. REM episodes tend to get longer in the second half part of the night sleep, while stage 3 occurs earlier in the first half of the night [19]. Finally, there are several times of short wakefulness throughout the night, after the initial onset of sleep.

tion models) can be built on personal webpage datasets. In particular, the All-Kth-Order Markov model has been found to be most effective for web user request prediction [21]. Furthermore, due to the complexity of user interests, Markov models are being used to keep track of user interests in terms of navigational behaviors [22].

Last but not least, speech recognition, the automatic process of translating spoken words into text is widely explored in computer science [23]. Modern general speech recognition systems are based on Hidden Markov Models (HMM) [24]. As illustrated in Figure 1.2, the standard output should be a sequence of symbols or quantities [25]. Human speech can be extracted as a Markov model for many stochastic purposes: a speech signal can be approximately thought of as a piecewise stationary signal in a short time scale, and then a Markov model can be trained automatically and applied easily with good time performance. In addition to hidden Markov models, linear dynamical systems (LDE) use a continuous state variable with linear Gaussian dynamics and measure [26]. LDEs have

Table 1.1: A sample of user behavior sequences. Each sequence corresponds to a user's requests at the level of page category [20]. Web users switch from one category to another (itself inclusive) after staying on it for a certain period of time.

User	Behavior Navigation Sequences						
1	frontpage	news	travel	travel			
2	business	business	business	sports	sports	sports	sports
3	tech	weather	weather	news			
4	news	local	news	weather	news		
5	sports	sports	sports	sports	sports	sports	sports
6	bbs	bbs	misc	misc	misc		
7	opinion	opinion	misc	health	health	health	
8	on-air	on-air	bbs	bbs			
9	news	news	news				
10	weather	sports	sports	sports			

also been used in practice on the same type of applications. For instance, a switching linear dynamical system is used for jointly modeling the dynamics of both the raw speech signal and the raw noise [27].

1.2 Motivation

As significant symbolic signal analyses, all of the above three examples share some data characteristic: scarcity of occurrences of specific dynamical state transitions over the entire time series. In sleep medicine, a hypnogram is defined to be a sequence of sleep stages [28], which consists of sleep bout durations and sleep stage transitions. A key characteristic of these sleep sequences is a few number of dynamical event occurrences in sleep data. For instance, there may be only a few transitions from stage 2 to rapid eye movement (REM) stage or from REM stage to wake stage throughout an entire all-night hypnogram [17]. In web user navigation log data, each sequence corresponds to a user's web requests. Switches from one category of webpages to another one rarely happen in sequences. Imagine that a user may click on webpage links about sport topic and stay on them looking for interesting news. During the time interval, there is no action of switching

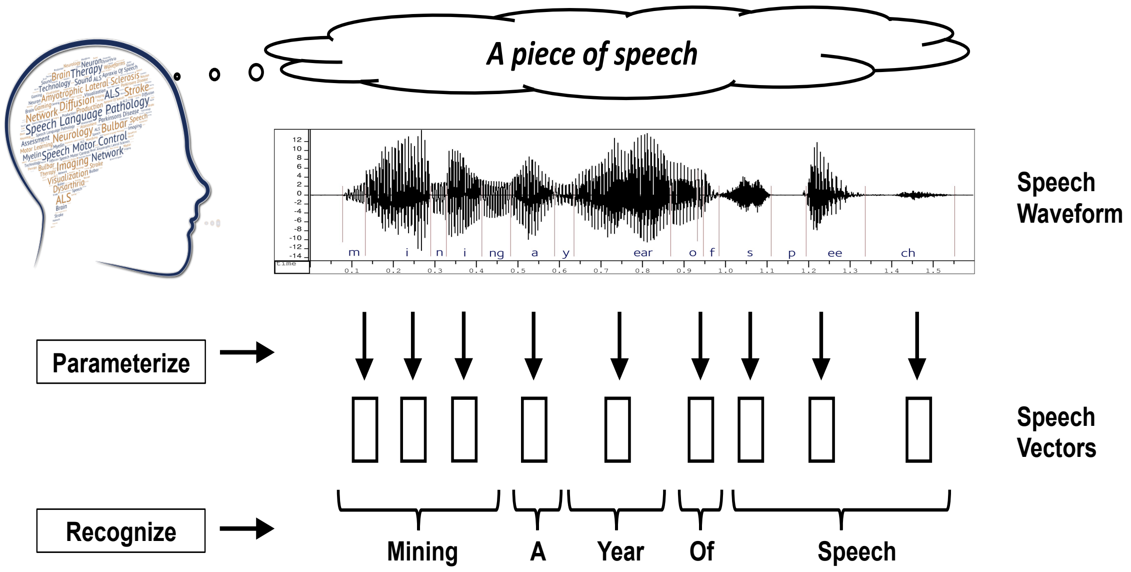


Figure 1.2: A sample diagram of speech recognition. Personal voice is stored as speech waveform in a computer. To apply a hidden Markov model (HMM) for speech recognition, a sequence of speech vectors representing a word are used as observations in a HMM. A part of speech waveform corresponding to one word is derived from a sequence of discrete hidden states [25].

topics until the user changes his or her focus. Lastly, in human speech surveys, there is a common scenario that persons keep silence for a certain amount of periods to collect their thoughts before answering questions in questionnaires. The fluctuant speech signal probably leads to a few number of state transitions from the silent period to the voluble one. All of these situations are classified as the scarcity of dynamical events coming out in a sequence of states.

However, the above phenomenon where changes occur infrequently makes it too hard to build stable dynamical models on temporal data sequences, since there is a big variation of modeling results over individual sequences. Take an artificial case using a synthetic dataset as an example. In Figure 1.3, two clusters with different colors are a collection of mixture data produced by uniformly distributed selection between two generative hidden Markov models (HMM): $HMM1$ and $HMM2$ (see section 2.4), each with two hidden states. Two-state transition probabilities are fully determined by the main diagonals in transition matrices as follows:

$$HMM1 : Transition\ Matrix = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} \quad Emission\ Matrix = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

$$HMM2 : Transition\ Matrix = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \quad Emission\ Matrix = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}.$$

To simplify the visualization of *HMM* on two-dimensional plane in Fig. 1.3, values along the main diagonals in the HMM transition matrices are considered. Here, cluster center $(0.9, 0.9)$ and $(0.7, 0.7)$ are representative of two *HMMs* respectively. Each generated sequence in a cluster has its own *HMM* built on itself and is coordinated by the main diagonal values in the two-state transition matrix. Sparse distributions of data points (depicted by coordinates) around cluster centers verify the underlying assumption that there exists big variations of individual models built on sequences individually, let alone variations of two generative models (e.g., *HMM*1 and *HMM*2 here) with individual models over individual sequences. In reality, for example, sleep sequences present a challenge to modeling algorithms, in that key dynamical events such as stage transitions occur very sparsely within a hypnogram, making it hard to extract reliable dynamical information from a single night of sleep of a given individual [9].

Therefore, in this dissertation, we are interested in automated algorithms [29] for the discovery of dynamical patterns in sequences with seldom occurrences of dynamical events. The following section introduces an original framework of simultaneous modeling and clustering over time series on which we work. It also describes potential challenges in the framework and presents approaches to tackling those challenges.

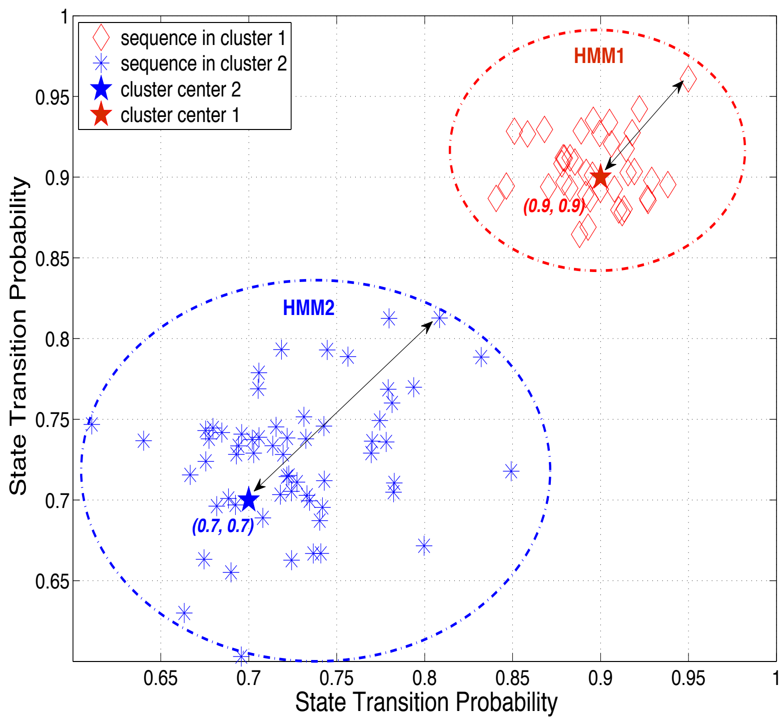


Figure 1.3: Variances of *HMMs* on temporal sequences with infrequent dynamic events in two clusters. The double-arrow line measures the distance between a cluster center and data points within the cluster. Cluster centers represent generative *HMM* models; data points represent individual models built on individual sequences. The length of a double-arrow line indicates the variation of models over sequences within the same cluster.

1.3 Research Challenges Addressed in This Dissertation

To discover sequential patterns using dynamical state-based models, researchers have proposed different approaches. Several of them used clustering of models. [30] presented Markov chains with an agglomerative clustering procedure. [31] grouped web users using a set of first-order Markov chains under the Expectation-Maximization framework. [32] generalized the mixture model (e.g., hidden Markov model) approach to clustering in feature space. However, some of these approaches are highly dependent on modeling individual instances. For example, [30] needed to build up a Markov chain model over each instance, and [33] optimized the parameters of the fitting function for each data instance. All the above methods have shortcomings when dealing with sparsity of dynamic events in sleep studies, since small amounts of temporal information in each time series are not sufficient for reliable statistical modeling of sleep [9].

The collective dynamical modeling-clustering (CDMC) algorithm [34] addresses the problem of scarcity of dynamical events by pooling data across multiple time series, simultaneously partitioning the collection of time series by dynamical similarity. CDMC reduces the model variation by selectively aggregating instances through clustering. This reduction in variation is accomplished in CDMC without the loss of detail that would result by simply aggregating data without regard for dynamical similarity. The result of CDMC is a collection of groups of instances such that instances within a given group are dynamically similar, while instances in different groups are not. The following is an outline of the CDMC procedure as shown in pseudocode in Algorithm 1:

- An initial grouping of instances $x_1, \ldots, x_n$ into k clusters is provided (step 1-2).
- The grouping is iteratively refined by repeating the following steps until the similarity of two successive clusterings (e.g., c and c_{old}) reaches a predetermined similarity threshold *minSim* (step 3):
    - In LEARNMLPROTOTYPES step, a maximum likelihood dynamical model M_i is built from each cluster C_i (step 5).
    - In LEARNMLCLUSTERLABELS step, each instance x is assigned to the cluster $C(x)$ for which the generative likelihood $P(x|M_{C(x)})$ is maximized (step 6).
- A final clustering c of the dataset and a generative model M_i for each of the clusters are returned (step 7).

Algorithm 1 Collective Dynamical Modeling-Clustering (CDMC) [34]

```
Input: An unlabeled time-series dataset $D = \{x = (a_i(x)) | i = 1, 2, \ldots, n\}$; a positive integer, $n$, the number of instances in $D$; a positive integer, k, for the desired number of clusters; an initial guess $c_0 : D \to \{1, \ldots k\}$ of the cluster label $c_0(x)$ of each instance $x \in D$; parameter values, $s$, specifying the desired configuration of the models (e.g., number of states); and a real number $minSim$ between 0 and 1 for the minimum clustering similarity required for stopping.
Output: A set $M_1, \ldots M_k$ of generative dynamical models (with configuration parameters $s$), together with a cluster labeling $c : D \to \{1 \ldots k\}$ that associates to each data instance, $x$, the index $c(x)$ of a model $M = M_{c(x)}$ for which the generative likelihood $\prod_{x \in D} P(x|M_{c(x)})$ is as high as possible.
CDMC($D$, $k$, $c_0$, $s$, $minSim$)
1. $c(x) = c_0(x)$ for all $x$ in D
2. $c_{old}(x) = 0$ for all $x \in D$
3. while CLUSTERINGSIMILARITY(c, $c_{old}$) < minSim
4.     $c_{old} = c$
5.     $(M_1, \ldots M_k)$ = LEARNMLPROTOTYPES($D$, $k$, $c$, $s$)
6.     $c$ = LEARNMLCLUSTERLABELS($D$, $M_1, \ldots M_k$)
7. return $M_1, \ldots M_k$, $c$
```

Note that the details of cluster initialization, dynamical model type, and similarity metric are left unspecified in the general version of CDMC. [34] includes an illustration that uses pseudorandom initialization, hidden Markov models, and the Rand index in these roles.

Collective dynamical modeling & clustering framework is a general algorithmic strategy: simultaneous clustering and dynamical modeling of sequence data [34] that is shown in pseudocode in Algorithm 1. However, it leaves open several important choices yet to be made during its development. This dissertation addresses the following three essential aspects of the CDMC framework:

1. Clustering Initialization Techniques

   The pseudorandom clustering initialization may fail to provide good cluster assignments for data instances, which may lead to a long iterative process of training dynamical models in CDMC. Therefore, it is worthwhile investigating other initialization techniques to produce better cluster guesses, for example, one examining selected statistical characteristics of time series or clustering time series based on similarity or distance metrics.

2. Dynamical Model Type

   Markov models were used as dynamical models in CDMC by [34]. However, due to inherent limitations of first-order Markov chains and Hidden Markov models (i.e., implicit exponential distribution), they are not expected to be a good fit for sequences with seldom-dynamical events. Since the degree to which CDMC can improve the capability of characterizing internal data structure depends on the choice of model, alternative models for analyzing temporal data with scarcity of dynamical events should be investigated. For example, Markov models with explicit duration distribution are able to explicitly specify distributions of state durations.

3. Convergence Property

   The convergence property in the CDMC framework should be examined both from a theoretical and an experimental point of view. The standard Rand index is used in [34] to measure the similarity of clusterings in two consecutive CDMC iterations. It greatly influences the rate of convergence, the speed at which clustering results approach its local or global optima. Hence, other metrics of evaluating clusterings need to be investigated, such as the adjusted Rand index that corrects for clustering agreements due to chance, information-theoretic measures such as normalized mutual information, and so on.

In addition to the above outstanding problems, scalability has become an equivalently significant factor when dealing with big volumes of sequence data. A systematic deployment plan for distributing the CDMC algorithmic framework across various computation units will be very valuable for practical applications in future.

In sum, the final objective of this dissertation is to significantly strengthen the collective dynamical modeling and clustering (CDMC) of time series with goals of effectiveness, efficiency, robustness, and scalability. Figure 1.4 presents a sketch of targets in this dissertation. The following section describes the approach taken to address these targets.

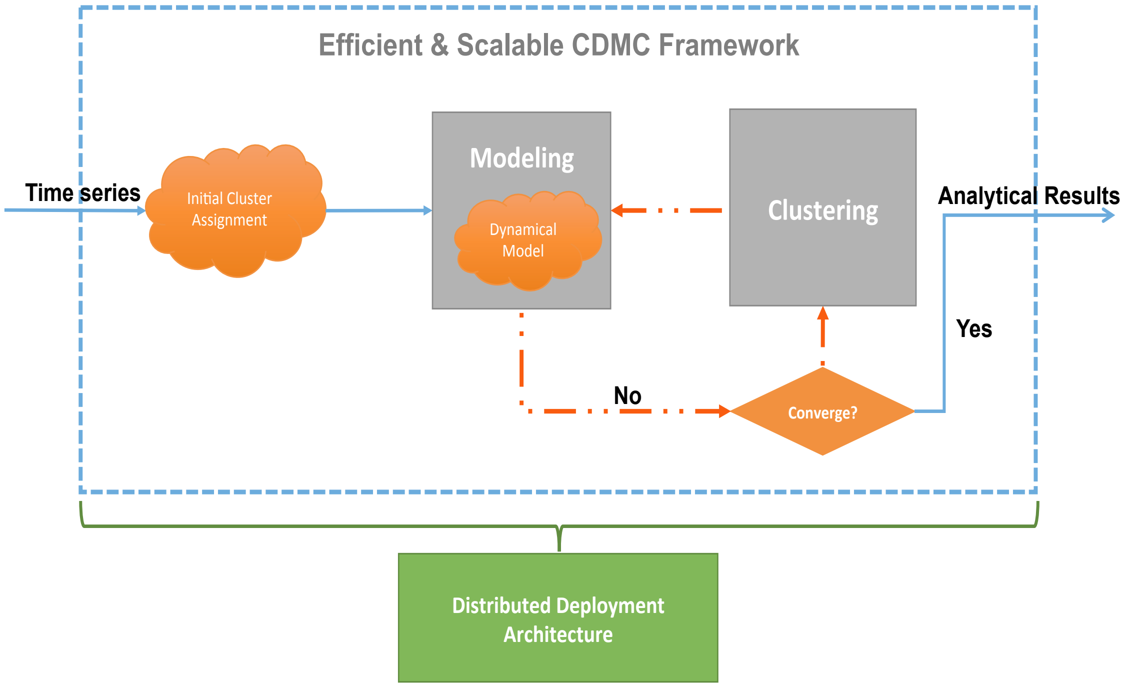

Efficient & Scalable CDMC Framework
Time series
Initial Cluster Assignment
Modeling
Dynamical Model
Clustering
Analytical Results
Yes
No
Converge?
Distributed Deployment Architecture

Figure 1.4: Efficient & Scalable CDMC Framework. The grey regions include the original version of the CDMC framework in Algorithm 1; the orange regions indicate the work done in this dissertation to improve the CDMC algorithm; the green region is an option for dealing with large-scalable time series in a distributed system introduced in this dissertation.

1.4 Proposed Solutions

Investigations concerning initialization, dynamical model type, and convergence have been conducted both in theoretical and practical ways. The following is a summary of the research contributions in this dissertation:

- To cluster discrete time series with scarcity of dynamical events, we first propose novel Dynamic Time Warping (DTW) variants that penalize deviations of the warping path from the path of constant slope. This overcomes the over-warping issue, while retaining the efficiency advantages of approaches based on global constraints, and without relying on domain dependent user input as in variable penalty DTW. We propose these modified DTW metrics for clustering initialization within the combined dynamical modeling-clustering (CDMC) framework [35] on discrete time series in Chapter 2. Clustering experiments over synthetic data as well as over human sleep data show that the proposed methods yield significantly improved accuracy and generative log likelihood as compared with standard dynamic time warping.

- To model the dynamics of symbolic representations of time series, we examine first-order Markov chain and hidden Markov model [15]. We also propose the use of flexible and expressive semi-Markov chains in Chapter 3 when modeling temporal sequences with sparse occurrences of dynamical events. The superiority of semi-Markov chains over Markov chains relies on the ability of the former to explicitly specify various distributions of state bout durations, rather than implicitly assume an exponential distribution as is the case in traditional Markov models. We show that semi-Markov chains provide better clustering results in Chapter 3.

- To study convergence in CDMC, we provide a basic theoretical proof of the convergence property of the CDMC framework in Chapter 4. We investigate the effects of our efficient initialization technique and our selected dynamical models on the CDMC convergence. The resultant significant differences in paired t-test confirm the advantages of those improvements. We also explore stopping criteria, as another factor with regards to convergence, to evaluate clustering agreements. Rand index (RI), adjusted Rand index (ARI), and normal mutual information (NMI) are included in this investigation.

- To accommodate big data in the CDMC framework, we address the crucial need to scale up the original CDMC framework in a distributed environment. The core components of our distributed approach include grouping incoming data instances and modeling over the dataset. We use Storm [36], an open source distributed real-time computation system, to support batch and distributed processing of data. A systematic evaluation of the proposed approach is carried out via experimentation with real application data on human sleep and web user navigation behavior to establish the efficiency and scalability of our approach.

1.5 Dissertation Organization

The rest of the dissertation is organized as follows. Chapter 2 presents a solution to the initialization problem mentioned in section 1.3, experimental results and analyses. Chapter 3 proposes a suitable model to capture the dynamics of symbolic time series, and provides experimental results. Chapter 4 studies the convergence property of the CDMC framework both from a theoretical and from an experimental point of view. Chapter 5 investigates a systematic way of deploying CDMC in a distributed environment. Chapter 6 illustrates the applicability of the extended CDMC framework to practical applications. Chapter 7 provides conclusions of the research in this dissertation, and Chapter 8 describes some future directions of this work.

Chapter 2

Clustering of Time Series

Time series mining has become pervasive in practice [37]. Applications of time series mining span online commerce [38], stock price prediction [39], human health [40], data summarization [41], and telecommunications [42]. Clustering [43] is an essential technique in time series analysis, especially when input data are unlabeled [44]. Moreover, given the widespread use of discretization of time series, such as in text mining, bioinformatics, and other applications [7], clustering of symbolic data is a central topic in data mining.

Figure 2.1 is a state-of-the-art summary of time series clustering techniques based on distinct manners of manipulating time series [44, 45]: there are totally four types of clustering approaches over time series: raw-data based (the first data flow on top), feature-based (the second one), and model-based (third & fourth) methods [44]. If raw data (time series) serve as input to clustering procedures, classical clustering approaches could be applied with similarity metric to evaluating time series. Otherwise, a reduced representation of time series, by way of feature extraction or sequence discretization, becomes an input to clustering processes. In this dissertation, data source is symbolic dataset and we focus on modeling sequences of symbols, which is categorized into the

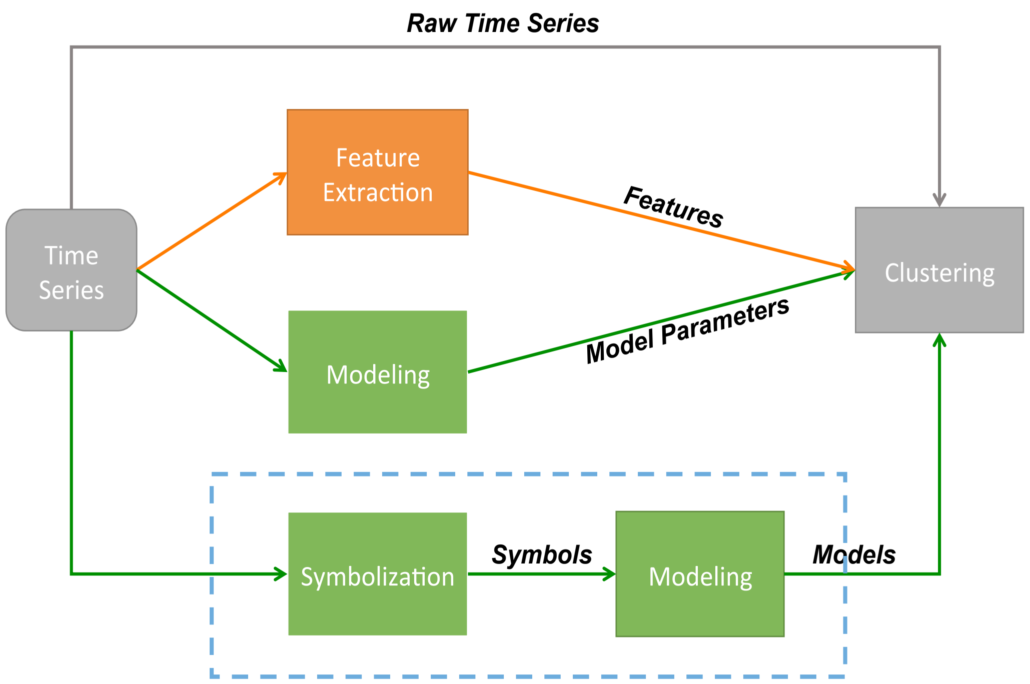


Figure 2.1: Four Time Series Clustering Approaches [44]. (1) clustering over raw time series, (2) clustering over extracted features of time series, (3) clustering over modeling raw time series, (4) clustering over modeling symbolic time series. Note that the area surrounded by dashed line is the clustering approache we work on in this dissertation.

fourth one.

Regardless of a variety of time series clustering algorithms in Figure 2.1, one of crucial components in them is to define a measure of similarity in time series [46]. For traditional distance metrics, they are used to evaluate real-valued time series. However, inherent limitation of Euclidean distance, for example, is sensitivity to distortion in time axis, especially in time series of different lengths [47]. For symbolic sequences, traditional distance measures only could be exploited if each symbol is encoded as a value or if distance metric between symbols is designated well. Even worse, discrete data have unique data characteristics, such as few dynamical events, long durations of a given state, and big variances of individual models on sequences in Fig 1.3.

On the other hand, the rise of dynamic time warping (DTW) metric [48] stems from stretching or shrinking a certain range of data points between time series to align them in a non-linear manner. Thus, dynamic time warping overcomes known difficulty of Euclidean distance [49], and leads to massive applications. The definition of cost function in DTW reduces the trouble of transforming symbols into values. However, it does not emphasize the importance of discrete data features. Furthermore, DTW metric has become an important way of measuring distances between time series [50, 51]. Theoretically, DTW maps one time series to another in a nonlinear transformation by minimizing the warping distance between the two. It was first introduced in speech recognition and gradually involved in various analytical tasks concerning medicine [52], bioinformatics [53], entertainment [54], etc.

In this chapter, we first introduce standard dynamic time warping in section 2.1. Then, we apply dynamic time warping for clustering initialization in CDMC, as a similarity metric in hierarchical clustering in section 2.2. Next, we propose variants of deviated dynamic time warpings in section 2.3. Finally, we analyze experimental results in section 2.4.

2.1 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) is a dynamic programming algorithm that provides an optimal alignment between two time series by nonlinearly warping their time dimensions [49]. DTW has been extensively used in speech recognition, periodic movement capture, and other areas [55, 56]. In this dissertation, DTW is used as a measure of similarity for unsupervised clustering of time series.

The following are the essentials of standard dynamic time warping technique, as described in [57].

We consider two time series $X = (x_1, x_2, \ldots, x_N)$ of length $N \in \mathbb{N}$ and $Y = (y_1, y_2, \ldots, y_M)$ of length $M \in \mathbb{N}$, with individual values x_i, y_j in some feature space $\mathcal{F}$. The essential notions are listed as follows:

- **Local Cost Measure.**

  A local cost measure is a function

$$c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0} \tag{2.1}$$

  The value of $c(x_i, y_j)$ is small if $x_i \in \mathcal{F}$ $(i \in [1:N])$ and $y_j \in \mathcal{F}$ $(j \in [1:M])$ are close to each other, and otherwise not. For discrete time series, one can use a cost matrix to define the values $c(x,y)$ for all pairs of values; the simplest possibility is to use the identity matrix, that is, to let $c(x,y)$ = 0 if they are the same, otherwise $c(x,y) = 1$.

- **Cost Matrix.**

  A cost matrix is a matrix

$$C \in \mathbb{R}^{N \times M} \tag{2.2}$$

  which consists of local cost measure for each pair of elements between sequence X and sequence Y. Note that $C(n,m) = c(x_n, y_m)$. For discrete time series, it is usually a matrix of 0*s* and 1*s*.

- **Warping Path.**

  A warping path between X and Y is a sequence

$$p = (p_1, p_2, \ldots, p_L) \tag{2.3}$$

  where $p_l = (n_l, m_l) \in [1:N] \times [1:M]$ for $l \in [1:L]$ (any path starting at bottom-left point and ending in the top-right point in Figure 2.2). $\max\{N,M\} \leq L \leq N+M$. The warping path must satisfies the following three conditions:

  - *Boundary condition*: $p_1 = (1,1)$ and $p_L = (N,M)$ at the start and end points respectively.
  - *Monotonicity condition*: horizontal and vertical components increases monotonically: $n_1 \leq n_2 \leq \ldots n_L$ and $m_1 \leq m_2 \leq \ldots m_L$.
  - *Step size condition*: for each $l \leq L$, the difference $p_{l+1} - p_l$ is one of (1,0), (0,1), (1,1).

- **Total Cost.**

  The total cost of a warping path p between X and Y ($\Phi_p(X,Y)$) is

$$\Phi_p(X,Y) = \sum_{l=1}^{L} c(x_{n_l}, y_{m_l}) \tag{2.4}$$

  where $c(x_{n_l}, y_{m_l})$ defines local cost measure in equation 2.1, which maps data point at n_l in sequence X to data point at m_l in sequence Y.

- **Optimal Warping Path.**

  An optimal warping path p^* is one having minimum total cost $\Phi_{p^*}(X,Y)$ among all warping paths from p_1 to p_L. In typical dynamic time warping computation, it takes dynamic programming optimization to compute the optimal warping path (see details in [57]). In the context of clustering in section 2.2, $\Phi_{p^*}(X,Y)$ is referred to as the *DTW distance* between sequences X and Y.

The standard dynamic programming approach to DTW for input sequences of length n implicitly considers all pairings of time indices in any of two input sequences. This computation leading to $O(n^2)$ time complexity. In the past decades, researchers had struggled for speeding up calculation of warping paths on 2-dimensional search space (cost matrix) in Figure 2.2, where a pair of time series are laid along x and y axes respectively. The approaches make DTW computation more efficient and generally fall into two types [58]:

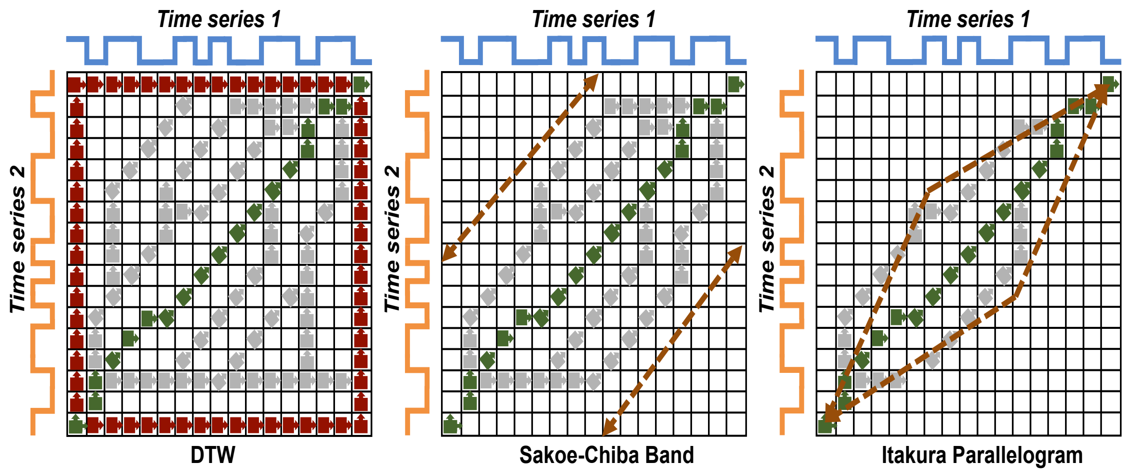


Figure 2.2: DTW and its variants. (1) DTW [49], (2) DTW with Sakoe-Chiba band [55], (3) DTW with Itakura Parallelogram [56]. If (2) or (3) is set in the cost matrix, a portion of areas (top-left and bottom-right ones) are eliminated; only central parts along the diagonal line are taken to speed up the computation of the minimum cost path (the green one starting from bottom-left corner and ending in top-right corner). Such operation guards against pathological warping paths like two paths indicated by red squares with directional arrows in the leftmost subplot, which results from the decreased areas of searching optimal warping paths. In a conclusion, under two global constraints of bands, search scopes for dynamic time warping are reduced significantly $(1) \rightarrow (3)$).

1. **Constraints** - Imposing global constraints, adjusting step size condition, or adding local weight vectors in cost measures. It is based on an idea of reducing searching areas in the rectangle valued by cost matrix at the risk of acquiring suboptimal (possibly not optimal) warping path.

2. **Data Abstraction** - Calculating DTW on a coarsened version of time series rather than on a complete resolution of time series. It is based on the strategy of dimensionality reduction and still likely to have a suboptimal path due to the loss of information in original resolution.

Generally speaking, suboptimal constrained versions of Dynamic time warping aim to reduce the time complexity by restricting the portion of the index space considered in the warping search. As variants of DTW, consider Itakura parallelogram and Sakoe-Chiba band in Figure 2.2. Itakura slope-constrained DTW [56] imposes slope constraints on the warping path, thereby constraining the search for a warping path to a parallelogram in the index space. The Sakoe-Chiba approach [55] constrains the warping search to a band in the index space. An alternative hierarchical multi-resolution approach has the potential to reduce the time complexity of dynamic programming computation to $O(n)$ [58].

2.2 Dynamic Time Warping Clustering (DTWC)

Dynamic time warping has been applied to initialization of a different clustering technique based on Hidden Markov Models [51]. To give a basic cluster guesses of time series, DTW-initialized clustering [59] is proposed as preprocessor of input data to CDMC.

The core clustering initialization approach is described in pseudocode in Algorithm 2. This approach performs agglomerative metric clustering using the distance function computed by DTW. The main steps of the proposed DTWC initialization are:

- Place each instance $x_1,...,x_n$ in its own cluster $C_1,...,C_n$ (step 1-2)
- Repeat until there are only k clusters left (step 3)
  - Merge the closest clusters, C_i and C_j; the distance measure between two instances (i.e., x_s and x_t) is defined by DTW; the distance measure between two clusters is the average distance of instances in these clusters (step 4-5)
- Return the final partition of the dataset D into k clusters (step 7)

Constrained DTWC (cDTWC). A fast variant of DTWC, constrained DTWC (cDTWC), is obtained by using Itakura slope-constrained DTW instead of standard DTW when computing the distance metric in Algorithm 2. The reduced search area over warping space saves the computation time and finds an optimal warping path. The two initialization techniques for CDMC will be compared with pseudorandom initialization in section 2.4.

Algorithm 2 DTW-Driven Clustering (DTWC)

```
Input: An unlabeled time series dataset D = $\{x_1, x_2, \ldots, x_n\}$; a positive integer, k, for a preassigned number of clusters; a predefined local distance measure $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ where $\mathcal{X}$ denotes the feature space in which the $x_i$ take their values.
Output: A partition $C$ of $D$ into $k$ clusters
DTWC($D$, $k$, $d$)
1. $C_i = \{x_i\}$ for each $x_i$ in D
2. $m = n$
3. while $m > k$
4.     $(i^*, j^*) = \arg\min_{i,j \in \{1,\ldots,m\}} \bar{d}(C_i, C_j)$
           $= \arg\min_{i,j \in \{1,\ldots,m\}} \left\{ \frac{\sum_{l(x_s)=C_i, l(x_t)=C_j} \mathbf{DTW}(x_s, x_t, d)}{|C_i| \cdot |C_j|} | x_s, x_t \in D \right\}$
5.     Merge $C_{i^*}$ and $C_{j^*}$ so that $C = \{1, \ldots, m-1\}$
6.     $m = m - 1$
7. return $\{C_1, \ldots, C_k\}$
```

2.3 Deviated Dynamic Time Warping

Despite promising results in section 2.4, standard DTW as a similarity measure for unsupervised clustering of time series suffers from certain problems. One of these is is time complexity. Variants of DTW (See Figure 2.2) have been proposed that focus on improving efficiency by globally constraining the warping path to a predefined geometric region such as Sakoe-Chiba band [55] and Itakura Parallelogram [56]. They imposed constraints on searching path space in a global sense, which limited how far the warping path may be away from the diagonal path between two time series. However, they both neglected progressively varied segments of warping path areas around the line, which potentially gave a more precise description of difference in time series. Furthermore, they failed to distinguish different time series, for which the overall distribution of sequences seems similar to each other to some degree but actually not. Even worse was that the duration of each discrete value and the transitions between them in time series were permitted to vary.

An additional concern with DTW as a measure for unsupervised clustering is the over-warping problem shown in Figure 2.8. Over-warping refers to unnatural alignment of dissimilar segments in two time series. In Figure 2.8, a subsequence of length over 300 in one patient is matched by dynamic time warping to a subsequence of length less than 10 in another. The result is unacceptable, yet the standard dynamic time warping distance between the two segments is zero. The reason causing such bad scenario is that a typical segment has long duration of a state and infrequent state transition, which makes it easy for original dynamic time warping to excessively extend or shrink a segment of one sequence to align another sequence, known as over-warping problem. Explicitly penalized DTW has been developed to address over-warping. For example, [60] proposes variable penalty DTW, which reduces non-diagonal moves during alignment. However, this approach is heavily dependent on a user-defined penalty function and thus difficult to apply in practice.

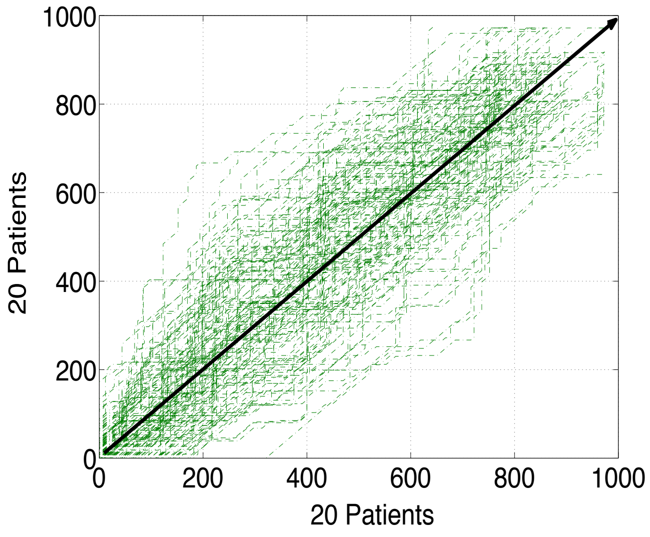


Figure 2.3: Optimal time warping paths (dashed lines) between 20×20 pairs of human sleep recordings. The best warping paths are usually close to the path of constant slope (solid diagonal line) from bottom-left to top-right in the warping space. The varying boundary of the distribution suggests the desirability of adaptively identifying warping areas locally instead of using a predefined global constraint.

Regardless of the above issues, many puzzles still surround standard dynamic time warping, which wasted much research effort [61]. One confusion is that a wider band in DTW searching space produces a better classification accuracy. Some researchers [1] have argued that the effect of warping band width on the quality of the results is greatly domain dependent and that a narrow band might be valuable. The distribution of optimal warping paths between pairs of sleep time series in Figure 2.3 likewise suggests that the use of local search constraints would be desirable.

In a word, it is necessary to provide a novel way of calculating optimal warping path, not only emphasizing locally concentrated searching area but also automatically penalizing non-diagonal alignments (not strongly associated with domain knowledge). In other words, it must make senses that a large amount of areas in warping space does not need to be examined during the process of finding the optimal warping path between time series. Moreover, the benchmark of evaluating the degree of deviation of warping path in diagonal direction should be as much as possible to be independent of users' experience.

To face the above challenges, we propose two versions of a modified dynamic time warping approach for comparing discrete time series. This approach is motivated by the observation that the distribution of dynamic time warping paths between pairs of human sleep time series in Figure 2.3 is concentrated around the path of constant slope. Both versions use a penalty term for the deviation between the warping path and the path of constant slope for a given pair of time series. In the first version, global weighted dynamic time warping, the penalty term is added as a post-processing step after a standard dynamic time warping computation, yielding a modified similarity metric that can be used for time series clustering. The second version, stepwise deviated dynamic time warping, incorporates the penalty term into the dynamic programming optimization itself, yielding modified optimal warping paths, together with a similarity metric.

Experiments over synthetic data in section 2.4, as well as over human sleep data in section 6.1.2, show that the proposed methods yield significantly improved accuracy and generative log likelihood as compared with standard dynamic time warping.

2.3.1 Deviation Measure

As stated above, we address the standard dynamic time warping concerns of over-warping and time complexity, by penalizing non-diagonal moves in the search for an optimal warping path. Based on empirical observations discussed below, we find out that a deviation plays an important role in achieving the goal.

Intuitively, deviation originates from an observation that the boundary of warping space inclines to be close to the diagonal line in the space (see Figure 2.3). Another observation is that for two sequences with a small dynamic time warping value, they may be unexpectedly different from their overall distributions along time axis. Therefore, the deviation aims at adjusting optimal warping path according to progressive variations of the current path with respect to its distance to the diagonal one. Given two optimal warping paths with the same total cost value, one with smaller deviation is preferable to the other one.

Formally, **deviation** refers to the area Δ_p (see shaded areas in Figure 2.4) surrounded by

- The warping path p for two time series;
- The diagonal path of constant slope.

Deviation is computed by the procedure described in Algorithm 3. The following are the main steps:

- Initialize the deviation Δ_p to be 0. (step 1)
- Calculate the slope k and the intercept b of the diagonal path defined as $y = k * x + b$. (step 2-3)

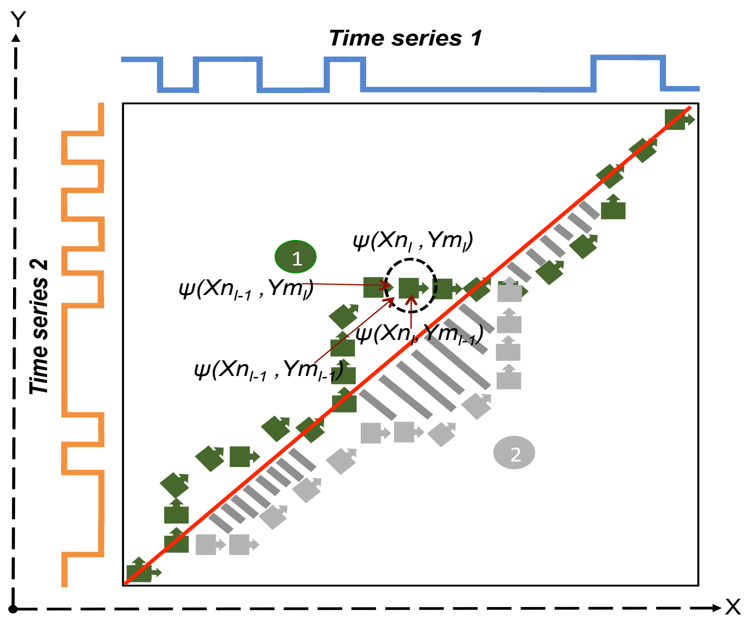


Figure 2.4: Deviation (e.g., gray shaded area) of warping path (squares with directional arrows) from path of constant slope (solid red line). Given two warping paths (1 and 2), the path with smaller deviation (the green one) is better. For stepwise deviated DTW, path 1 is generated by stepwise deviated dynamic time warping, where the next move to $\Psi(X_{n_l}, Y_{m_l})$ is determined by adjacent cells: $\Psi(X_{n_{l-1}}, Y_{m_{l-1}})$(diagonal move), $\Psi(X_{n_{l-1}}, Y_{m_l})$(right move), and $\Psi(X_{n_l}, Y_{m_{l-1}})$(up move).

- Repeat until the end point p_L is reached. (step 5)

  - Add the absolute vertical distance between Y axis between the current point p_i and the previous point p_{i-1} to the deviation Δ_p, if they differ horizontally. (step 6-7)

  - update the counter i. (step 8)

- Return the deviation Δ_p (step 9)

Algorithm 3 computes the corresponding deviation after an optimal warping path is found. This algorithm works as a post-processing step for global weighted dynamic time warping in section 2.3.2. When computing areas in deviation, we count differences of y value in both the optimal warping path and the diagonal path under the same x value.

Algorithm 3 Deviation Calculation

Input: An optimal warping path $P = \{p_1, p_2, \ldots, p_L\}$; L : the total number of points in the path, P.

Output: The deviation Δ_p of p from the straight line through p_1 and p_n in warping space.

ComputeDeviation(P)

```
1. $\Delta_p = 0$
2. $k = (M-1)/(N-1)$
3. $b = M - k * N$
4. i = 2
5. **While**$(p_i \neq p_L)$
6.    **if**$(n_i \neq n_{i-1})$
7.        $\Delta_p = \Delta_p + |m_i - (k \cdot n_i + b)|$
8.    $i = i + 1$
9. **return** $\Delta_p$
```

2.3.2 Deviation-Based Dynamic Time Warpings

We propose two versions of deviation-based dynamic time warpings: global weighted dynamic time warping (gwDTW) and stepwise deviated dynamic time warping (sdDTW). gwDTW penalizes standard DTW distance (total cost of optimal warping path) in terms of the deviation area relative to the optimal warping path as described in section 2.3.1. On the other hand, sdDTW calculates the deviation by each step when constructing the warping path and integrates the deviation into the process of determining which direction the current step should be moved to. In addition, due to the linearity of DTW distance, the square root of deviation should be more compatible with it, in contrast to the surface of non-sqrt deviation. The experiments in section 2.4 validate this claim.

Global Weighted Dynamic Time Warping: The global weighted dynamic time warping

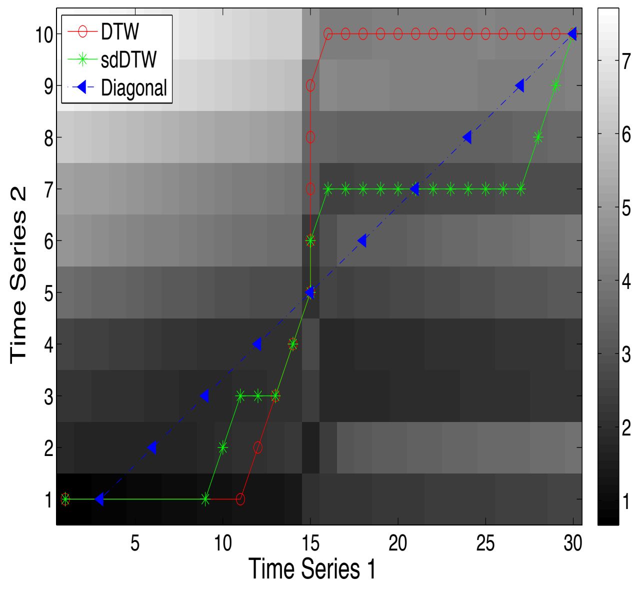


Figure 2.5: Optimal warping paths derived from standard DTW and stepwise deviated DTW (sdDTW) for two time series. Standard DTW allows large deviations in searching for a warping path of minimum total cost. Over-warping occurs at the beginning and end of the path shown. sdDTW aligns time series closer to the diagonal line. Background shading indicates local cost in sdDTW, which increases with distance to the diagonal line.

($gwDTW$) distance between sequences X and Y is defined as

$$\begin{aligned} gwDTW = {} & \lambda_{gw} \cdot \Phi_{p^*}(X,Y) \\ & +(1-\lambda_{gw}) \cdot \sqrt{\Delta_{p^*}(X,Y)} \end{aligned} \tag{2.5}$$

where a penalty term Δ_{p^*} is added as a post-processing step following standard dynamic time warping computation, between two sequences X of length N and Y of length M. λ_{gw} controls the balance between traditional dynamic time warping cost and the deviation $\Delta_{p^*(X,Y)}$ described in section 2.3.1. λ_{gw} ranges from 0 to 1. The best value of λ_{gw} is determined empirically by accuracy of correctly grouping similar sequences to one cluster in experiments (see section 2.4). Higher the accuracy of clustering sequences is, better λ_{gw} is. The square root of the deviation $\Delta_{p^*(X,Y)}$ is used because it scales linearly with Euclidean distance.

Stepwise Deviated Dynamic Time Warping: The stepwise deviated dynamic time warping ($sdDTW$) distance between sequences X and Y is defined as

$$sdDTW = \Psi_{p^*}(X,Y) \tag{2.6}$$

where $\Psi_{p^*}(X,Y)$ is the minimum total cost of a warping path p^* (see Figure 2.5) obtained by replacing the local cost measure in Equation 2.1 by the modified measure in Equation 2.7.

Technically, stepwise deviated dynamic time warping incorporates a penalty term (with respect to the deviation of current position (x,y) in the warping path from the diagonal path for two sequences X and Y) into dynamic programming optimization itself, yielding modified optimal warping paths. The followings are modified versions of basic concepts in standard dynamic time warping (see section 2.1):

- **Local Cost Measure.**

  A modified local cost measure is defined as:

  $$\varphi(x,y) = \lambda_{sd} \cdot c(x,y) + (1-\lambda_{sd}) \cdot \sqrt{\Delta(x,y)} \tag{2.7}$$

  where $c(x,y)$ denotes the original local cost measure in equation 2.1. $\Delta(x,y)$ is the deviation of a position (x,y) relative to the diagonal path of constant slope for sequences X and Y. See section 2.3.1. λ_{sd} is a parameter that determines the relative weights of the standard local cost measure in equation 2.1 and the deviation in this position.

- **Total Cost.**

The modified total cost of a warping path p between X and Y is

$$\Psi_p(X,Y) = \sum_{l=1}^{L} \varphi(x_{n_l}, y_{m_l}) \tag{2.8}$$

where $\varphi(x_{n_l}, y_{m_l})$ is the modified local cost measure in equation 2.7 mapping data point at n_l in sequence X of length N to data point at m_l in sequence Y of length M.

- **Optimal Warping Path.**
  A modified optimal warping path p^* is one having minimum total cost $\Psi_{p^*}(X,Y)$ among all warping paths.

We use dynamic programming as in standard DTW to compute the modified optimal warping path, based on the **modified accumulated cost matrix**

$$\bar{D}(n,m) = \Psi(X(1:n), Y(1:m)) \tag{2.9}$$

where $X(1:n) = (x_1, \ldots, x_n)$ and $Y(1:m) = (y_1, \ldots, y_m)$. $n \in [1:N]$ and $m \in [1:M]$. That is $X(1:n)$ and $Y(1:m)$ are subsequences of X and Y. The procedure is as follows:

- Initially, $\bar{D}(n,1) = \sum_{k=1}^{n} \varphi(x_k, y_1)$ and $\bar{D}(1,m) = \sum_{k=1}^{m} \varphi(x_1, y_k)$.
- Iteratively, take the minimum of accumulated cost matrix from three immediately adjacent directions: $\bar{D}(n-1,m)$, $\bar{D}(n,m-1)$, $\bar{D}(n-1,m-1)$.
- Until the final position (N,M) is reached, $\bar{D}(N,M)$ is the optimal dynamic time warping distance with respect to stepwise deviated dynamic time warping.

Note that both global weighted dynamic time warping and stepwise deviated dynamic time warping could be degenerated into standard version of dynamic time warping if weight values (λ_{gw} and λ_{sd}) were set to be 1. In practice, they could probably find the same warping path even if weight values (non-zero) take effects between local cost measure and deviation computation.

Algorithm 4 Deviation-Based DTW Clustering (dDTWC)

```
Input: An unlabeled time series dataset $D = \{X|X \text{ is a time series}\}$; a positive integer, $k$, the desired number of clusters; a predefined local distance measure $d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}$ where $\mathcal{F}$ denotes the feature space in which the time series in $D$ take their values.
Output: A partition of $D$ into $k$ clusters
dDTWC($D$, $k$, $d$)
  1. for each $i$, let $C_i$ = a cluster that contains only the $i$-th time series in $D$
  2. $m$ = the number of time series in $D$
  3. while $m > k$
  4.     $(i^*, j^*) = \arg\min_{i,j\in\{1,\dots,m\}} \bar{d}(C_i, C_j)$ (where $\bar{d}$ is mean distance between instance pairs in the two clusters)
         $= \arg\min_{i,j\in\{1,\dots,m\}} \left\{ \frac{\sum_{X=C_i,Y=C_j} \mathbf{dDTW}(X,Y,d)}{|C_i|\cdot|C_j|} | X, Y \in D \right\}$
  5.     Merge $C_{i^*}$ and $C_{j^*}$ to reduce the number of clusters to $m-1$
  6.     $m = m - 1$
  7. return $\{C_1, \dots, C_k\}$
```

Deviation-based dynamic time warping clustering (dDTWC) performs unsupervised agglomerative hierarchical clustering of time series using the deviation-based DTW approaches to calculate distances. The proposed approach is shown in pseudocode in Algorithm 4 by utilizing **gwDTW** and **sdDTW** as distance metric. The main steps are:

- Initially each time series instance X is in its own cluster (steps 1-2).
- Repeat until only k clusters remain (steps 3-6):
    - Merge the closest clusters, C_i and C_j; the distance between two instances (X and Y) is defined by **gwDTW** or **sdDTW**; the distance between two clusters is the average distance between pairs of instances.
- Return clustering of dataset in k clusters (step 7).

$$\bar{D}(n,m) = min\{\bar{D}(n-1,m), \bar{D}(n,m-1), \bar{D}(n-1,m-1)\} + \varphi(x_n, y_m) \qquad (2.10)$$

Algorithm 5 Generation of Synthetic Data

Input: Positive integers N and L, semi-Markov chains $S_1, \ldots, S_k$.
Output: A collection of N sequences, each of length L, generated by a mixture of the semi-Markov chains $S_1, \ldots, S_k$.

```
genData($N$, $L$, ($S_1, \ldots, S_k$))
  1. $C = \{\}$
  2. for $i = 1, \ldots, N$
  3.     $j$ = random choice among $\{1, \ldots, k\}$
  4.     $x_i$ = sequence of length $L$ generated by $S_j$
  5.     $C = C \bigcup \{x_i\}$
  6. return $C$
```

2.4 Experimental Evaluation

The goal of this experimental study is to investigate performances of DTW-driven and Deviation-based clustering on discrete time series. We will show that: (1) Dynamic time warping clustering significantly improve grouping discrete time series. (2) Deviation-based clustering methods lead to significantly better clustering results than DTW in unsupervised clustering of discrete time series.

2.4.1 Experimental Setup & Methodology

Each experiment involved 100 trials. Wilcoxon-Mann-Whitney statistical hypothesis testing was used for comparison of medians. All experiments were performed in MATLAB® (*The MathWorks*, 2012). Unless stated otherwise, all variants of dynamic time warping were tested through simulation that used synthetic datasets. Before going through experimental details, it is necessary to describe synthetic datasets, evaluation and experimental protocols.

- **Synthetic Mixture Dataset.**

  In order to generate a sequence of a given length, L, from the mixture model, a random choice is first made between the two given models. A state sequence of the desired length is then generated by the randomly selected model. The random chain selection and state sequence generation process continues until a desired total number of sequences, N, has been generated. See Algorithm 5. In the experiments, $N = 100$ sequences, each of length $L = 100$, were used in all trials involving synthetic (semi-Markov or hidden Markov) data.

- **Semi-Markov Mixture Data.**

  A semi-Markov mixture dataset is generated from two distinct semi-Markov chains, each with two states, but with different transition probability matrices and state duration statistics. The use of synthetic data provides precise control over the generative statistical parameters. The following transition matrices and Weibull (shape, scale) parameter values (in that order) were used for the two semi-Markov chains. There were two Weibull distributions per chain, one for each state.

$$\textit{Semi-Markov chain 1:} \quad \begin{pmatrix} 0.90 & 0.10 \\ 0.10 & 0.90 \end{pmatrix}, \quad (3,5), \ (2.5, 4,5)$$

$$\textit{Semi-Markov chain 2:} \quad \begin{pmatrix} 0.15 & 0.85 \\ 0.85 & 0.15 \end{pmatrix}, \quad (3,4), \ (2.5, 3,5)$$

- **Hidden Markov Mixture Data.**

  A hidden Markov mixture data is generated as in [34], from two distinct hidden Markov models, each with two states that emit two symbols with given emission probabilities. The two models differ in their transition probability matrices. Self-transition probabilities of 0.6 and 0.8 were selected. The probabilities of transitioning from one state to the other were 0.4 and 0.2 respectively as follows:

Hidden Markov Model 1: $\begin{pmatrix} 0.50 \\ 0.50 \end{pmatrix}, \begin{pmatrix} 0.60 & 0.40 \\ 0.40 & 0.60 \end{pmatrix}, \begin{pmatrix} 1.00 & 0.00 \\ 0.00 & 1.00 \end{pmatrix}$

Hidden Markov Model 2: $\begin{pmatrix} 0.50 \\ 0.50 \end{pmatrix}, \begin{pmatrix} 0.80 & 0.20 \\ 0.20 & 0.80 \end{pmatrix}, \begin{pmatrix} 1.00 & 0.00 \\ 0.00 & 1.00 \end{pmatrix}$

- **Cluster Validity.**

  Clustering quality is evaluated by comparing the clustering results to known class labels for the synthetic data described above. The class label of an instance is the model that generates that instance. The classification accuracy (fraction of instances that are classified correctly, relative to the set of all instances) is used as the evaluation metric. Higher classification accuracy of a clustering indicates a more meaningful data partition.

- **Statistical Significance.**

  Pairwise comparisons of median classification accuracy values of DTW-driven as well as deviation-based DTW clusterings against the accuracy of standard DTW clustering (in the case of synthetic data) and of negative log likelihood values of gwDTW and sdDTW clustering against that of standard DTW clustering (in the case of unsupervised clustering of human sleep data in section 6.1) were carried out by a non-parametric two-sided Wilcoxon rank sum test, since a Lilliefors normality test rejected normality at the $p < 0.05$ significance level in each case. A Bonferroni correction was performed jointly on the accuracy and log likelihood Wilcoxon p-values to ensure a familywise error rate less than 0.05.

- **Synthetic Data Classification Procedure.**

  Supervised classification was performed with the generating model label as the classification target. The goal of clustering [43] was therefore to partition the set of data instances into two groups that closely match the subsets of instances associated with the two generating models. The evaluation metric was classification accuracy. Statistical hypothesis testing was performed using a Wilcoxon rank sum test. An example of experimental procedure for deviation dynamic time warpings was as in the following pseudocode:

Experimental procedure, synthetic data classification:

```
begin
    for $i := 1$ to $TrialNum$
        SD = generateSyntheticDataset(N, L);
        Accuracy(1, i) = evaluateByDTW($SD$);
        Accuracy(2, i) = evaluateBygwDTW($\lambda_{gw}$, $SD$);
        Accuracy(3, i) = evaluateBysdDTW($\lambda_{sd}$, $SD$);
    end
    Perform Wilcoxon rank sum test over Accuracy.
end
```

Note that

- *generateSyntheticDataset* followed the description in *Hidden Markov Mixture Data*.
- *evaluateBy***DTW** refers to the clustering procedure in Algorithm 2 and clustering evaluation by classification accuracy. *evaluateBy***gwDTW** and *evaluateBy***sdDTW** likewise do.
- The total number of sequences, N, was set to be 100.
- The length of a sequence, L, was set to be 300.

– A predetermined number of trials, $TrialNum$, was set to 100.

2.4.2 DTW-Driven Clustering Evaluation

We compare DTW-based clustering and constrained DTW clustering with pseudorandom initialization as input to CDMC in experiments. The synthetic semi-Markov mixture dataset was used in the evaluation of initialization techniques. This allows the use of classification accuracy to evaluate performance of cluster validity. For the initialization experiments using DTW, the local cost measure was defined as that if a pair of elements in two sequences are same, the cost is 0, otherwise, 1.

Pseudo-random initialization for CDMC was compared with DTW-distance clustering and Itakura constrained DTW (cDTW) clustering initialization techniques for CDMC in section 2.2, as well as with the DTW and cDTW clustering results directly. Table 2.1 lists t-test results with statistical significance $(p < 0.05)$ of accuracies between CDMC-rnd and DTWC. Figure 2.6 shows the resulting accuracy values. Pseudorandomly initialized CDMC (median accuracy 0.82) is not significantly more accurate than the two DTW-only clustering techniques without CDMC (medians 0.76 and 0.75, for full and constrained DTW, respectively). However, the DTW-initialized CDMC (median accuracy 0.94) significantly outperforms all other techniques.

Table 2.1: T-test of accuracies between CDMC-rnd and DTWC

H value	P value	CI
1	1.4034e-10	[0.1089, 0.1985]

2.4.3 Deviation-Based DTW Clustering Evaluation

Deviation-based DTW clustering as described in section 2.3.2 was compared with clustering using the standard DTW distance metric. For all dynamic time warping computa-

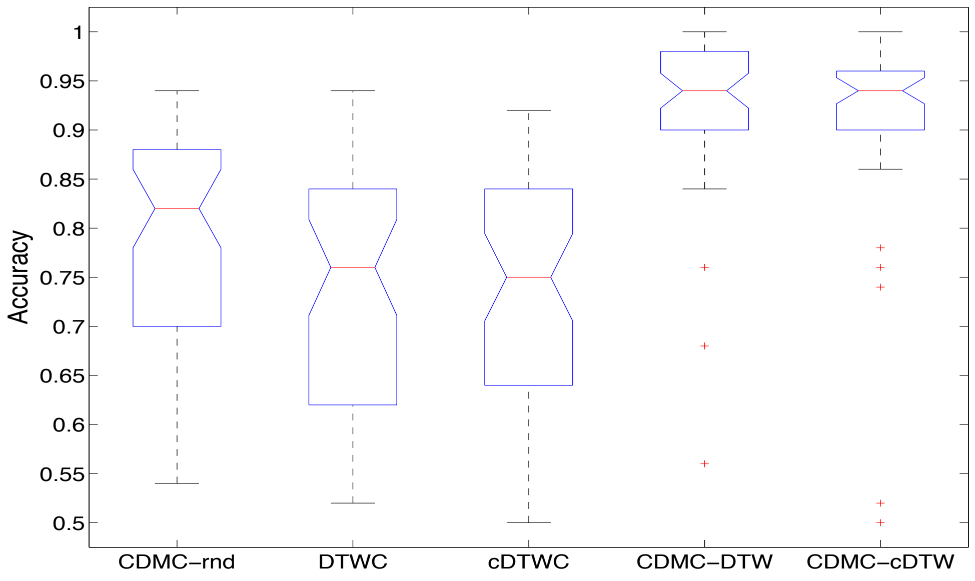


Figure 2.6: Accuracies for randomly initialized CDMC, DTW-only clustering (DTWC), constrained DTW-only clustering (cDTWC), DTW-initialized CDMC, and constrained DTW-initialized CDMC. Non-overlapping notches indicate significant difference in medians ($p < 0.05$). DTW-initialized and CDTW-initialized CDMC yield significantly better accuracies than the other clustering techniques.

tions, the local cost measure in equation 2.1 was defined as $c(x,y) = 1$ if the elements x and y are different, otherwise $c(x,y) = 0$. The weight values λ_{gw} and λ_{sd} in equations 2.6 and 2.5, respectively, were determined empirically in order to maximize mean accuracy over a sample of synthetic data (but separate from the synthetic data sample used for performance evaluation in Figure 2.7). λ_{sd} was set to 0.67. λ_{gw} was set to 0.83.

- *DTW-driven vs. Global Weighted DTW-based Clusterings*: We evaluate performance of clustering over synthetic data using globally weighted DTW (gwDTW) (Equation 2.5) as the similarity measure, as compared with standard DTW similarity.

  To determine the best weight value (e.g., λ_{gw}), an exhaustive search from 0 to 1 was conducted in the procedure of clustering over various synthetic dataset. The weight value with highest average accuracy was to be selected as λ_{gw}. 100 trials were performed to compare clustering performances of using DTW alone and global weighted DTW. They are compared to examine the effect of deviation areas on finding optimal path in the context of clustering. Clustering accuracies over the synthetic dataset are shown in Figure 2.7. gwDTW performs significantly better than standard DTW, proving the advantage of incorporating deviation into the dynamic time warping computation for clustering of synthetic time series data. Median accuracies appear in Table 2.2.

- *DTW-driven vs. Stepwise Deviated DTW-based Clustering*: We aim at comparing the use of standard dynamic time warping methods as similarity measure in clustering task with stepwise deviated versions of dynamic time warping.

  Similar to experiments using global weighted dynamic time warping, synthetic dataset was generated by two hidden Markov models as described before. The weight value λ_{sd} was empirically set to 0.67. Standard dynamic time warping and one adding deviation into DTW computation (sdDTW) were subjects. During clustering over synthetic dataset, accuracies for the three clusterings are shown in Figure 2.7. sdDTW significantly performs better than standard dynamic time warping alone in the Wilcoxon rank sum test. The medians of accuracies for the three clusterings in Table 2.2 strengthen the fact: deviation plays an impact on evaluating time series with long-duration state. The integration of DTW and deviation produces the best clusters according to accuracy.

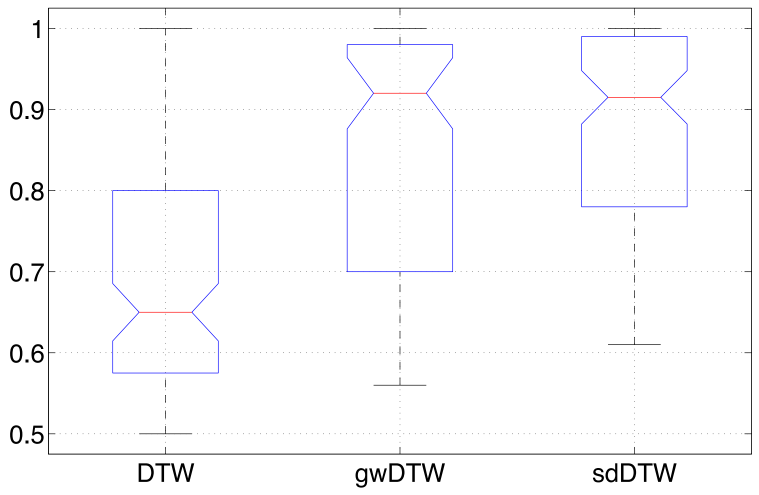
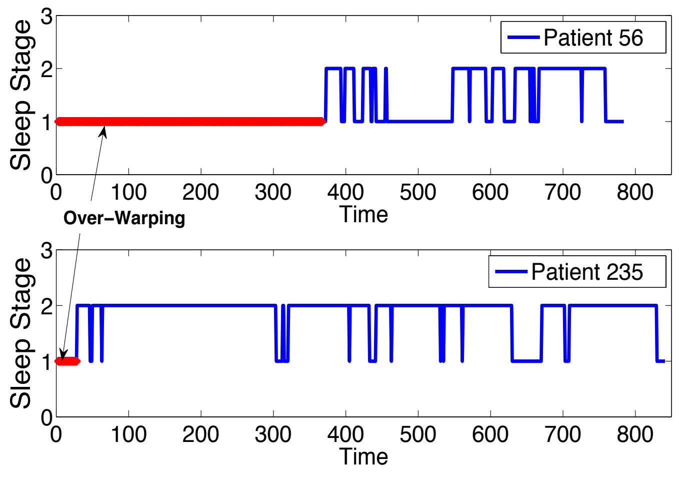


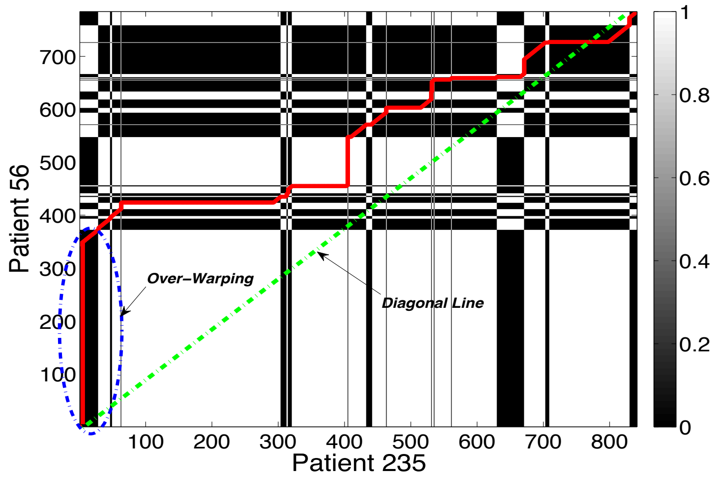
Figure 2.7: Clustering accuracies using standard DTW, gwDTW, and sdDTW as similarity measure over hidden Markov mixture data. Non-overlapping notches indicate significant difference in medians ($p < 0.05$). gwDTW and sdDTW are significantly more accurate than standard DTW.

Table 2.2: Median accuracies of clusterings based on DTW, gwDTW, and sdDTW. Asterisks denote Bonferroni-corrected statistical significance of differences with standard DTW in Wilcoxon rank sum test ($p < 0.05$).

DTW	**gwDTW**	**sdDTW**
0.65	0.92*	0.91*



(a) Over-Warping Occurrence in Time Series



(b) Over-Warping Path in Dynamic Time Warping

Figure 2.8: Over-warping of sleep stage sequences using dynamic time warping. (Left) Use of standard dynamic time warping inappropriately matches dissimilar segments (a long one versus a short one) in two sequences. (Right) The same over-warping problem described in terms of warping search area. The area circled by the dashed line indicates a large deviation of the standard warping path from the diagonal path of constant slope. The bar graph on the right indicates local cost measure and the background in the right figure shows the local cost matrix of two discrete time series (patient 56 and patient 235) in dynamic time warping computation (see section 2.1).

Chapter 3

Dynamical Modeling over Time Series

Discrete time series is a typical type of time series, in which each datum is chosen from some predetermined non-empty finite set, taken at uniformly spaced time instants [62]. It is also called discrete symbolic data, for example, sleep stage sequences in human sleep [15] and a paragraph of words in text [63].

Discrete time series can be described as alternations process among states. That is, discrete time series refers to a progression, starting with a given state, then cycling in which states alternate, and finally ending in a certain state. During a sequential recording, every state exhibits short or long uninterrupted bouts as time passes by in it. Regardless of typical patterns in the sequence, the alternation details vary across individuals, which may be affected by underlying factors.

Different from static information in time series [64], for example, total length of the time series and percentage of a given state in sequential data, there are two vital indicators capturing dynamical information in discrete time series. One is state transition, transforming from one state to another one (itself inclusive). A transition matrix of all state transitions gives the probabilities associated with state change. The other one is state bout duration, counting the number of a given state in a maximal uninterrupted segment of the state within a given sequence. The distribution of a state bout duration includes frequencies of state durations of different length for the specified state. In addition, state bout duration distribution is commonly depicted by some distribution functions (e.g., probability mass function [65]). As stated in chapter 6, the internal dynamics [8] are widely used in applications to sleep-wake architecture [66], where exponential and power-law functions are proposed as parametric models for the distributions of wake and sleep bout durations [10]. In a word, both state transition and state bout durations form the basis of the data representation for dynamics in time series in this dissertation.

In this chapter, we first introduce Markov models in section 3.1, then depict the semi-Markov dynamical model version of CDMC in section 3.2, and finally present the application of semi-Markov dynamical models in real dataset in section 3.3.

3.1 Markov Models

Markov model has been a powerful technique applied extensively for sequential data in real applications [24]. The reasons behind the use of Markov model in discrete time series analysis are ascribed to three aspects. Firstly, Markov model is defined for modeling time series, especially natural for discrete time series [67]. In other words, each element in a sequence is considered as a state and a switch event from one element to another as a state transition. Secondly, Markov model gives full support of mathematical basis, in particular, statistics, thus providing probabilistic explanations in reality [68]. Last but not least, Markov models are able to provide practical methods of training parameters by themselves efficiently [69]. To sum up, it is a good candidate for training models to characterize dynamics (i.e., state transitions) in discrete time series.

Different from Markov models, linear dynamical system (LDE) uses a continuous state variable with linear Gaussian dynamics and measurement [26]. LDE could be used in sleep research, speech recognition, and user behavior pattern since under the same assumption that there are hidden variables with Markovian dynamics like HMM. A switching linear dynamical system is applied for jointly modeling the dynamics of both the raw speech signal and the raw signal noise [27]. For human sleep research, it consists of five discrete sleep stages. The stage transitions are natural state transition of Markov model with the reasonable Markov assumption. Therefore, Markov models (e.g., Markov chain and Hidden Markov model) are more popularly used in human sleep than other non-Markov state transition models.

3.1.1 Markov Chains

Among Markov models, first-order Markov chain model provides a dynamical model of transitions between states [70]. It is a particular kind of Markov model, in which all well-defined states are observable (that is, there are no hidden states). Take a 3-state Markov chain model in Figure 3.1 as an example. The Markov chain consists of three states (e.g., state 1, 2, and 3) together with a state transition matrix that provides the probability of transition between each pair of states (e.g., $P_{1\to1}$ or $P_{1\to2}$). State transitions in the Markov model occur with a fixed probability in every cycle of a standard clock that is shared by all of the states. The Markov assumption establishes that state transition probability depends only on the current state, independent of previous states in sequential data. As a result, the duration of visits to a given state in a Markov chain model has a geometric (discrete exponential) probability distribution. For instance, the probability of transitions from state 1 to itself before leaving for a different one amounts to $P^t_{1\to1}(1-P_{1\to1})$ in Figure 3.1.

In practice, Markov chains [24] have been used to model dynamics of time event like speech recognition. A simple time-homogeneous Markov chain was first applied in the sleep domain [14]. However, due to geometrically distributed state bout durations as shown in Figure 3.1, Markov chains (and more generally, hidden Markov models) may not

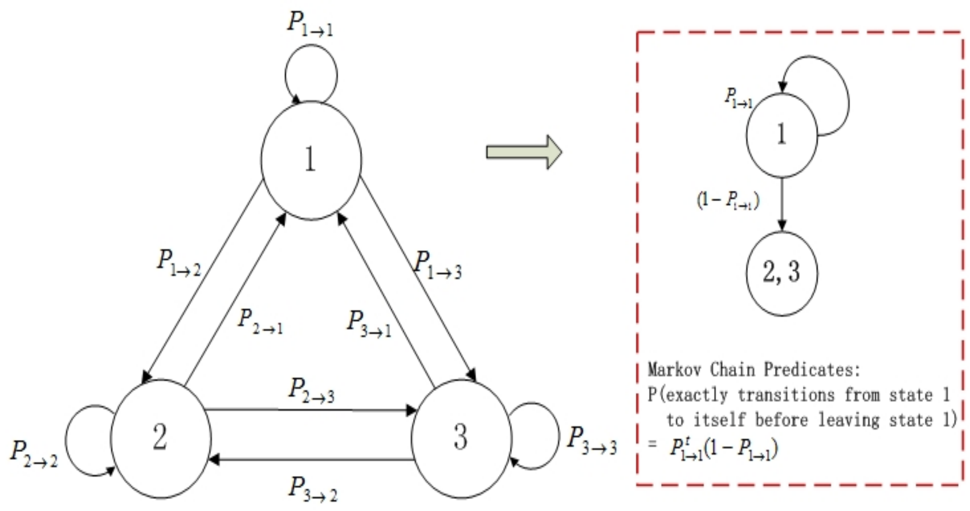


Figure 3.1: A 3-State Markov Chain with Exponential Decay. There is a probability $P_{i\to j}$ of transition from one state (e.g., S_i) to another (e.g., S_j) where $i, j = 1,2,3$. The probability of transitions from state 1 to itself before leaving for a different one amounts to $P^t_{1\to1}(1-P_{1\to1})$.

characterize distributions of state transitions more flexibly and accurately, contradicting known experimental observations [10, 15]. To overcome this limitation, we investigate semi-Markov chain models in section 3.2.

3.2 Semi-Markov Models

Experimental results [10, 16] demonstrate that geometric distributions are a poor fit for actual stage bout duration (see section 3.2.1) distributions of human sleep. For example, the wake stage has a distribution of bout durations with a slowly decaying tail that is more similar to a power-law function than to a discrete exponential. Motivated by this fact, our work employs a semi-Markov chain model for the sequence of sleep stages, instead of a Markov model.

Semi-Markov chains, as a variant of Markov chain models [24], are proposed to be more suitable for describing sequences with infrequent dynamical events, since they do not implicitly assume exponential distributions of state durations [16, 71]. The use of semi-Markov dynamical models in Collective Dynamical Modeling Clustering (CDMC) framework better describes the dynamic characteristics of human sleep as compared with first-order Markov chain models in section 6.

3.2.1 State Bout Duration Distribution

State bout durations form the basis of the data representation in time series. A state bout is defined as a maximal uninterrupted segment of the given state within a given sequence. The duration of a state bout is defined as the number of units which the bout spans. The frequency of a state bout duration is the number of state bouts of this same duration present in the sequence. The distribution of a state bout durations (that is, the frequencies of different bout durations for that state) can be depicted by the distribution function (probability mass function). As stated in section 3.1, [10, 11] have shown that good approximation to stage bout duration distributions can be obtained by using single exponential function and power law functions. To avoid implicit exponential distribution of state bout durations in Markov chains, exponential, power law, and Weibull density functions are fit to stage bout duration data:

- **Exponential Distribution.**

  The general form of a single exponential distribution [72] in Figure 3.2 is given in equation 3.1, where μ is the expected value.

  $$f(x;\mu) = \frac{1}{\mu} \exp \frac{-x}{\mu} \; (x \geq 0) \tag{3.1}$$

  The expected value of the exponential distribution for a given state in Maximum Likelihood Estimation (MLE) corresponds to the expected value of durations over the sequence data.

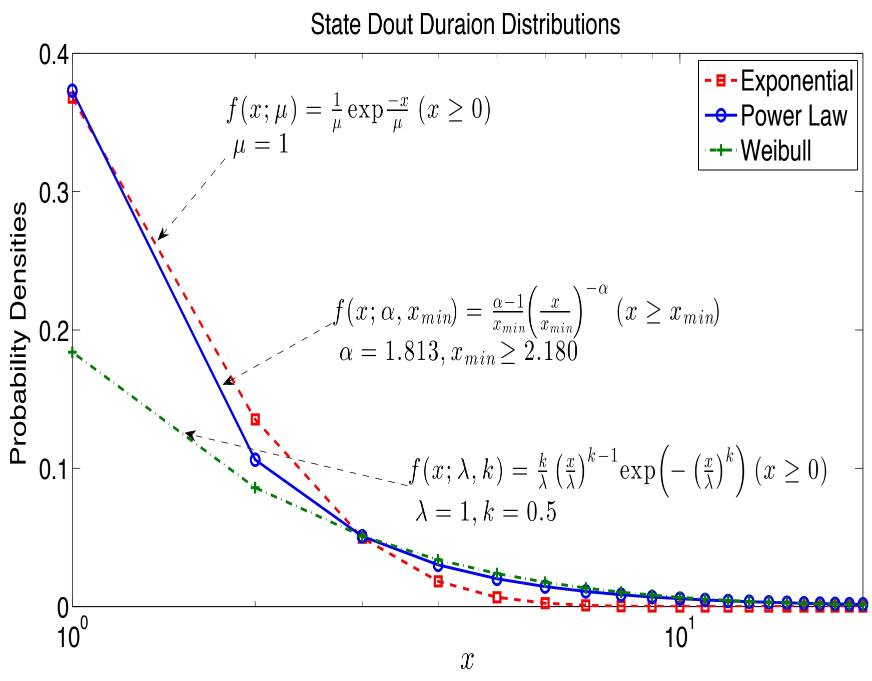


Figure 3.2: Example of state bout distributions. Exponential, power law, and Weibull ones are depicted in the range from 1 to 20. The exponential has the steepest change , the Weibull is the most smoothed one, and power law interpolates between them. Due to their different tailing characteristics in data, the three are fit to various state bout durations in input sequences. See experimental evaluation of sleep stages in section 3.3.3, for example.

- **Power Law Distribution.**

  The general form of a power law distribution [73] in Figure 3.2 is given in equation 3.2.

$$f(x;\alpha,x_{min}) = \frac{\alpha-1}{x_{min}}\left(\frac{x}{x_{min}}\right)^{-\alpha}\ (x \geq x_{min}) \tag{3.2}$$

  The estimation of α in the power law distribution using maximum likelihood estimation is shown in equation 3.3.

$$\alpha = 1+n\left[\sum_{i=1}^{n}\ln\frac{x_i}{x_{min}}\right]\ (x_i \geq x_{min}\ \alpha > 1) \tag{3.3}$$

  According to the estimation in equation 3.3, and based on the sequential data, x_{min}

should be 1, and a large enough range for initial values for α is the range from 1 to 10 times the MLE estimate of α. A delta increment value of 0.01 is used to select sufficient α values in this range. Note that n is the number of observed empirical data.

- **Weibull Distribution.**

  The general form of the Weibull distribution [74] is given in equation 3.4.

$$f(x;\lambda,k) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} \exp\left(-\left(\frac{x}{\lambda}\right)^{k}\right)\ (x \geq 0) \tag{3.4}$$

  where λ is scale parameter and k is shape parameter in the Weibull distribution. When $k = 1$, the Weibull distribution coincides with an exponential distribution (See Figure 3.3). The expected value of the Weibull distribution is given by equation 3.5.

$$E(x) = \lambda\Gamma\left(1 + \frac{k}{\lambda}\right) \tag{3.5}$$

  where Γ is the gamma function, which extends the factorial function. When $k = 1$, $E(x) = \lambda$. To be consistent with the kernel estimated data distribution, k is limited to a range between 0 and 2. λ can range from 1 to the maximum duration of any state.

To compensate for the scarcity of bout durations in the dataset, kernel density estimation [75] is applied. Kernel density estimation (KDE) is used for nonparametric probability density estimation. It is a useful statistical smoothing technique used when inferences about the population are to be made based on a finite data sample. The stage duration distributions calculated over discrete time series often contains many missing values for specific bout durations in states. KDE is used to smooth these data distributions, thus providing meaningful values for durations of different states as shown in Figure 3.4. A

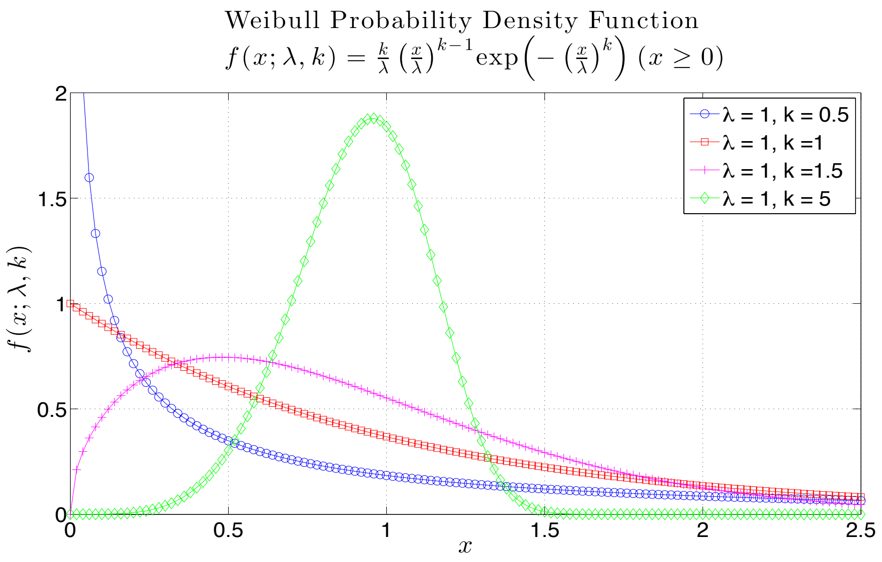
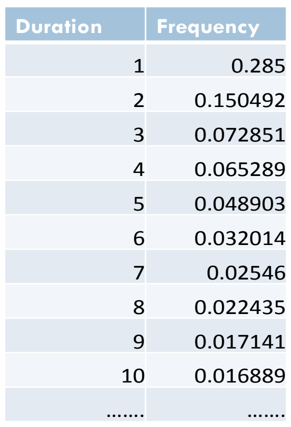


Figure 3.3: Weibull distribution family. If x is interpreted as a "time-to-failure" in statistical reliability theory, the Weibull distribution indicates a distribution where the failure rate is proportional to a power of time. The shape parameter k determines one of several cases: if $k < 1$, then failure rate decreases over time; if $k = 1$, then it does not change over time; if $k > 1$, then failure rate increases with time.

normal distribution with a kernel-smoothing window width of 1 was determined to produce the best results.

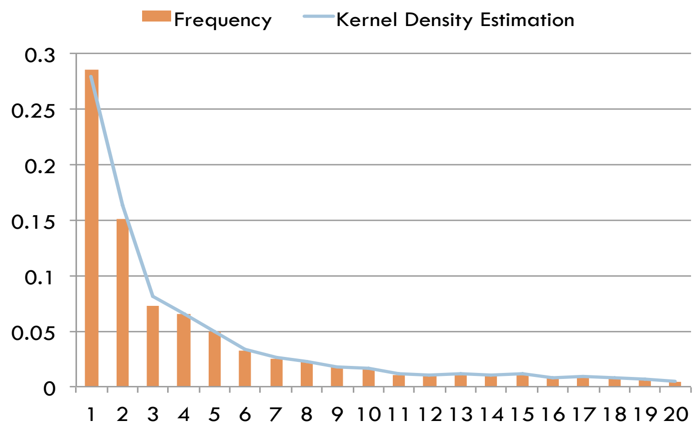
Once the state duration distributions have been smoothed using non-parametric kernel density estimation, parametric functions that closely approximate the shape of these duration distributions can be found. A new metric for evaluating the goodness of fit is introduced and used to select the best fit, which is similar to the Mean Square Error (MSE) metric applied to the logarithmically transformed frequency data [76], except that the parameter x appears in the denominator of the new metric. This new metric proved superior to other metrics (including MSE) in capturing the quality of the approximation as gauged by visual inspection during systematic experimentation.

$$GOF(x) = \sum_{k=1}^{n}\left(\frac{(\log s(x_k) - \log f(x_k))^2}{x_k}\right) \tag{3.6}$$



Duration	Frequency
1	0.285
2	0.150492
3	0.072851
4	0.065289
5	0.048903
6	0.032014
7	0.02546
8	0.022435
9	0.017141
10	0.016889
....	....

REM Stage



REM Stage

Figure 3.4: Kernel density estimation for sleep dataset. There are 244 patients: 122 males and 122 females all suffering from sleep problems. The frequency of stage bout durations for REM stage is shown in the left subplot. The kernel density estimation is used to smooth the frequency data and provides estimated values given stage durations in the right subplot.

In equation 3.6, s(x_k) refers to the nonparametric probability density estimation at duration x_k, and $f(x_k)$ refers to the fitting data on s(x_k). Logarithms are taken on the estimated data s(x_k) and fit data $f(x_k)$ to match the visual aspect of these values in a log-log scale plot.

The following approach is employed to search for the best possible curve fits for each state duration distribution.

1. Calculate the state duration distribution from sequences.
2. Smooth the calculated state duration distribution using KDE.
3. Fit exponential, power law, and Weibull functions to the smoothed state duration distribution using each of the following two approaches to estimate parameters:

   ML: Use Maximum Likelihood (ML) [77] to estimate the parameters of each of the fitting function families. As an illustration, the plots in Figure 3.6(c) depict the best ML approximation obtained for the REM stage duration distribution for each of the three fitting function families.

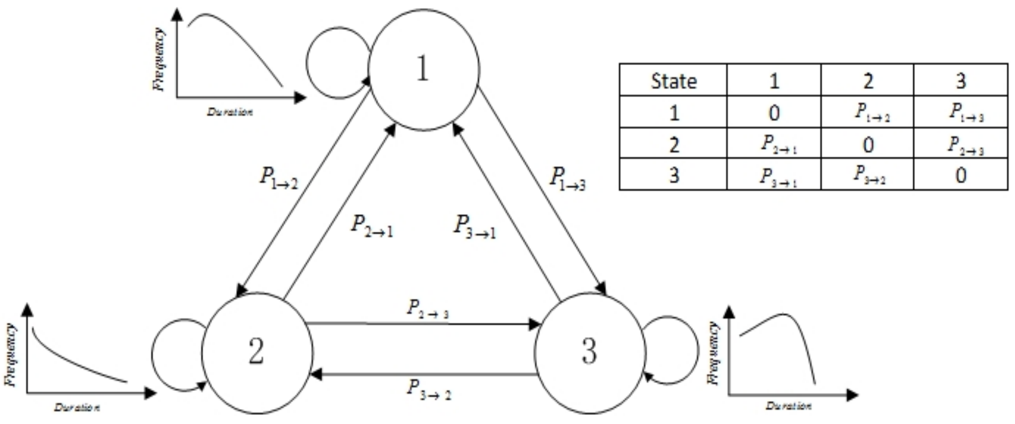
LS: Use Least Squares (LS) to estimate the parameters. In this case, the initial parameter values for the estimation were taken from the ranges described for each function family (exponential, power law, and Weibull). Repeat this for each possible initial point in the given range. Then use the GOF metric to select the best estimation obtained for each function family among all those obtained from the set of initial parameter values. The plots in Figure 3.6(c) depict the best LS fits obtained (together with the best ML fits).

4. Use the GOF metric to select the best estimation obtained by the ML and LS approaches above across all function families. Output this estimation as the best approximation for the state duration distribution.

3.2.2 Semi-Markov Chains With Explicit Bout Duration

In a semi-Markov model, the mechanism that determines the durations of state visits is independent of any model-wide clock. Each state i in a semi-Markov model of the type considered here has a specified visit duration distribution $P_i(\tau)$ in section 3.2.1; upon arriving in the given state, a random sample τ is taken from this visit duration distribution; the model then remains in state i for the duration τ, at the end of which time the next state is selected from among all states other than i, according to some Markov-type transition probability matrix with zeros along the diagonal. Other varieties of semi-Markov models also exist [78].

Different from first-order Markov chain in Figure 3.1, semi-Markov chains used as dynamical models in CDMC have two basic components: state transition matrix and state duration distributions for states. The state transition matrix is similar to that in the Markov



State	1	2	3
1	0	$P_{1\to 2}$	$P_{1\to 3}$
2	$P_{2\to 1}$	0	$P_{2\to 3}$
3	$P_{3\to 1}$	$P_{3\to 2}$	0

Figure 3.5: A 3-state semi-Markov chain model (SMM). Similar to a Markov chain model (MM), a SMM consists of a state diagram (which in this case includes three states: 1, 2, and 3) together with a transition probability matrix that provides the probability of transition between each pair of states. The difference between a SMM and a MM is that the probability of a self-transition is made equal to 0 in the SMM's transition matrix, and the duration of "staying" in that state is explicitly represented instead by the probability density function that approximates that state duration distribution.

chain, except that the probability of a transition from any state to itself is 0. The best fitting function selected by section 3.2.1 should be used as the description of the stage duration distribution of each stage. The mechanism of a semi-Markov chain model operates as follows in Figure 3.5:

1. Pick up a specific state, state 1, for instance;

2. Stay in the current state for a duration, randomly sampled from the duration distribution for that state (given by the approach of searching for best fitting function [15], expressed as $P_1(d_1)$);

3. Transiting from the current stage to the selected state according to the vector of state transition probabilities, $\{P_{1\to 2},\ P_{1\to 3}\}$.

4. Repeat step 2 and 3 indefinitely.

Since each state has its own bout duration distribution, semi-Markov models become a more flexible model than first-order Markov chain model. If distributions for all states in the semi-Markov chain were exponential, semi-Markov chain is degenerated into a Markov chain.

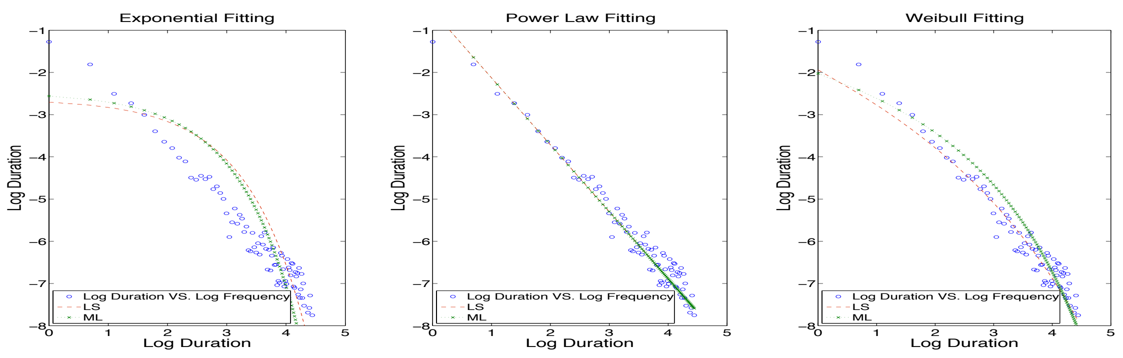
3.3 Experimental Evaluation

The goal of this experimental study is to present the superiority of semi-Markov chains over Markov chains and hidden Markov models for the purpose. We will show that (1) first-order Markov chains simply characterize features in discrete time series and (2) semi-Markov chains flexibly capture dynamical features of data and perform much better than Markov chains in modeling time series.

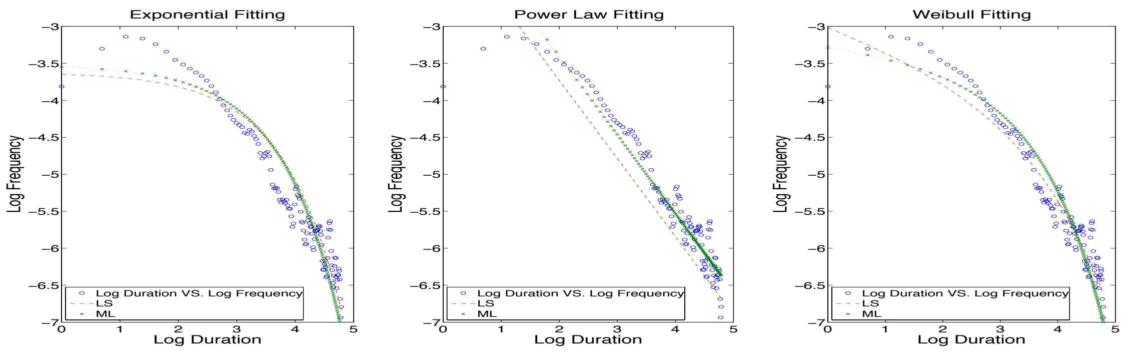
3.3.1 Experimental Setup & Methodology

We investigate the effect of the use of semi-Markov chains as dynamical models in CDMC, compared with first-order Markov chain in the following experiments. The relevant experimental setup includes:

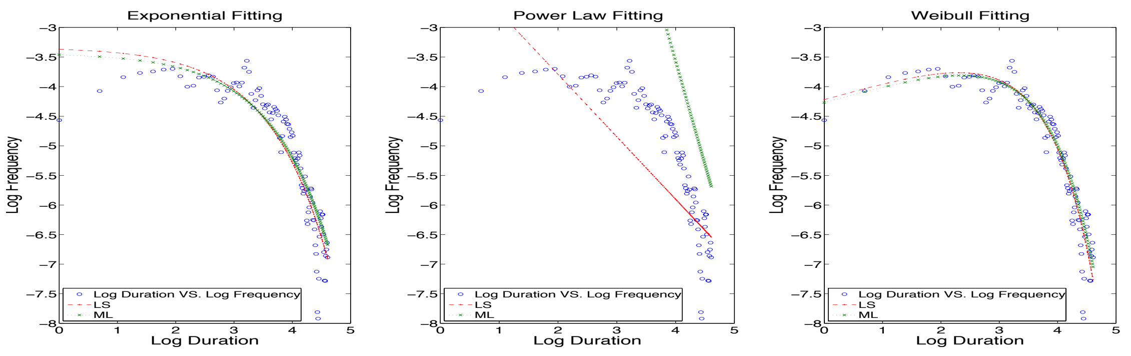
- **Human Sleep Sequences.**
  A real dataset of human sleep is manipulated in the experiment, in order to eliminate variances of synthetic dataset generated by three variants of Markov models, such as Markov chains (MM), hidden Markov model (HMM), and semi-Markov chains (SMM). The human sleep dataset consists of a total of 244 fully anonymized human polysomnographic recordings. They were extracted from polysomnographic overnight sleep studies performed in the Sleep Clinic at Day Kimball Hospital in Putnam, Connecticut, USA in section 6.1.1. Each polysomnographic recording is split into 30-second epochs and staged by lab technicians at the Sleep Clinic. Staging of each 30-second epoch into one of the sleep stages (wake, stage 1, stage 2,



(a) Wake Stage

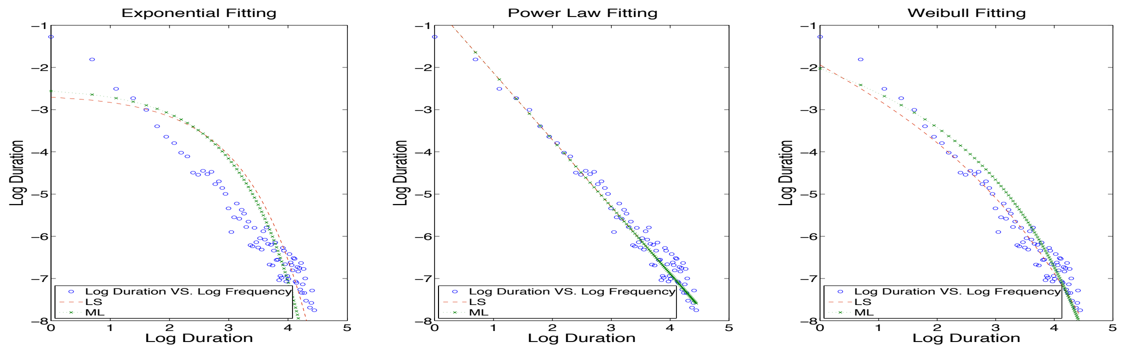


(b) NREM Stage

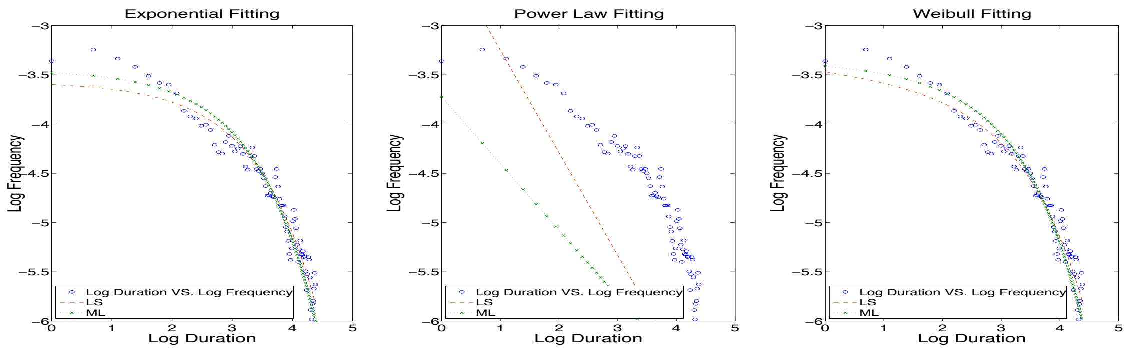


(c) REM Stage

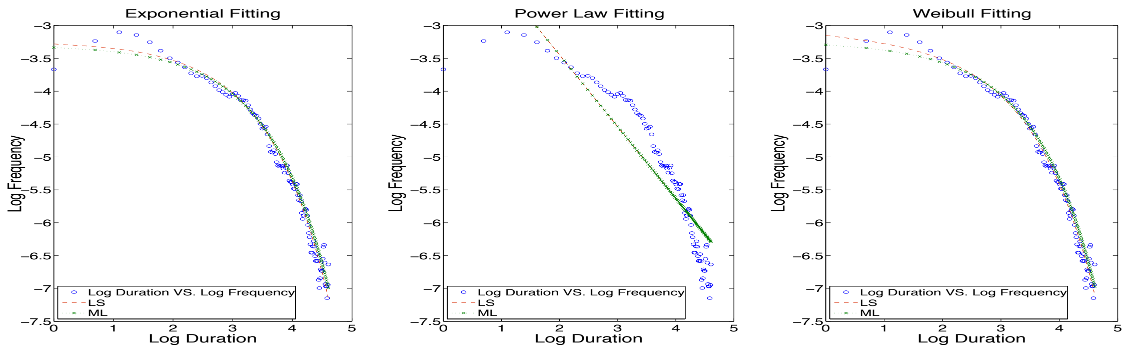
Figure 3.6: Curve fitting of sleep stage duration distribution in WNR dataset. The three grouped plots depict the fits obtained by using exponential functions (left), power law functions (center), and Weibull functions (right). In each plot, the results of the ML and LS approaches to find best fits are presented. For the REM stage, among all these candidate fits, the Weibull function achieves the best goodness of fit. The same are for the wake and NREM stages.



(a) Wake Stage

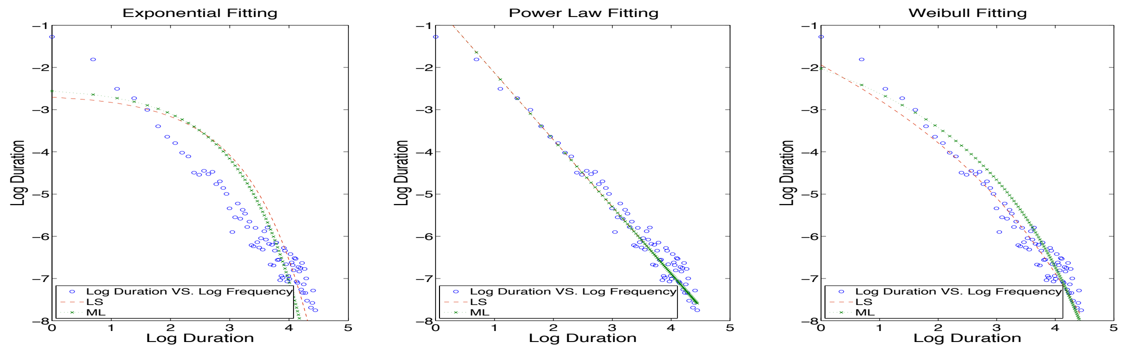


(b) Deep Stage

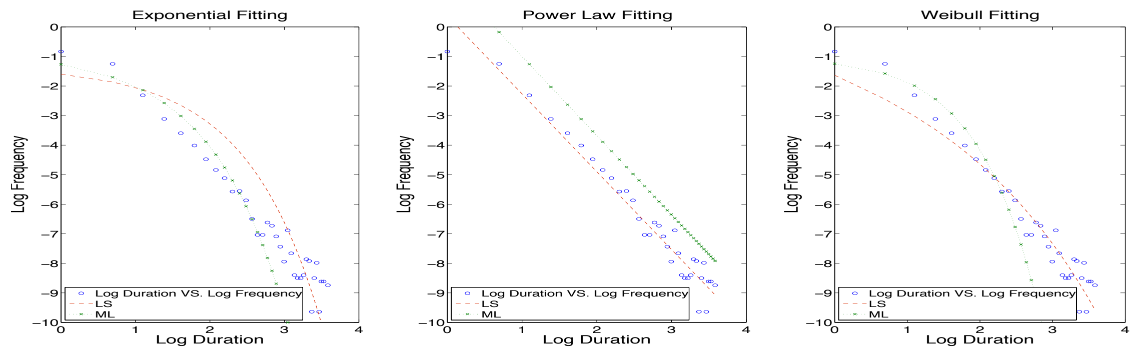


(c) Light Stage

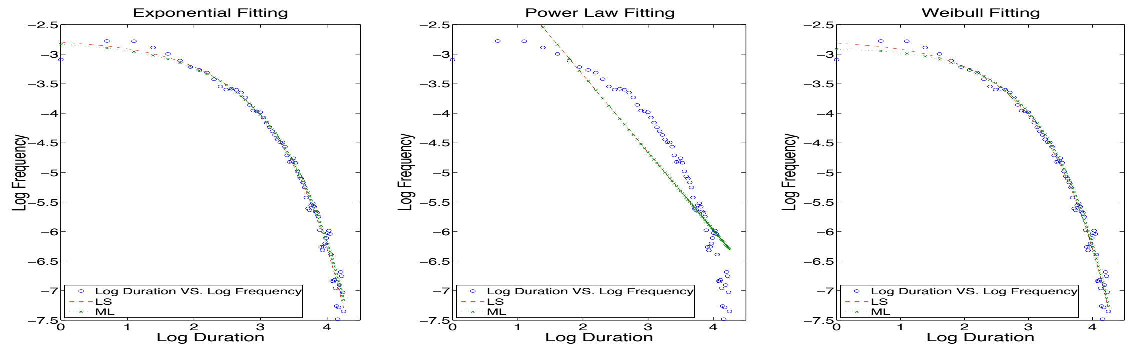
Figure 3.7: Curve fitting of sleep stage duration distribution in WDL dataset. The three groups of plots depict the fits obtained by using exponential functions (left), power law functions (center), and Weibull functions (right) in wake, deep sleep, and light sleep stages respectively. In each plot, the results of the ML and LS approaches to find best fits are presented. Among all these candidate fits, the Weibull function achieves the best goodness of fit for all three stages.



(a) Wake Stage



(b) Stage 1



(c) Stage 2

Figure 3.8: Curve fitting of sleep stage duration distribution in W5 dataset. The three groups of plots depict the fits obtained by using exponential functions (left), power law functions (center), and Weibull functions (right) in wake, stage 1, and stage 2 respectively. In each plot, the results of the ML and LS approaches to find best fits are presented. The Weibull function achieves the best goodness of fit in wake stage and stage 2. Power law performs best in Stage 1.

stage 3, and REM [18]) is done by analyzing EEG, EOG and EMG recordings during the epoch [79]. Stages 1, 2, and 3 are grouped into a non-REM stage, abbreviated as NREM. This condenses the representation of the sleep stages to three: Wake, NREM, and REM, collectively denoted WNR. Stage 1, 2, and REM could be combined into light stage and stage 3 as deep stage [19]. In the experiment, three different versions of the human sleep dataset are considered:

- (W5) uses the five standard stage labels Wake, 1, 2, 3, REM.
- (WNR) uses the three stage labels Wake, NREM (stages 1, 2, and 3), REM.
- (WDL) uses the three stage labels Wake, Deep (stage 3), Light (stages 1, 2, and REM).

- **Equilibrium Distribution.**

  We use equilibrium distribution to represent the large-time asymptotic probability of occupation of the various states in a Markov chain [80]. Therefore, the equilibrium distribution characterizes the long-term dynamics of discrete time series in a Markov chain model. The equilibrium distribution of a Markov chain (but not of a general semi-Markov chain) may be computed as the normalized eigenvector of the transition matrix with eigenvalue 1.

  To approximate the equilibrium distribution in a semi-Markov chain model, one can use simulation of the semi-Markov chain. Sufficiently long stage sequences are generated so that asymptotic probabilities of states can be estimated. These limiting values represent the equilibrium distribution of the semi-Markov chain model.

- **Fitting Function Difference.**

  In addition to the equilibrium distribution in the semi-Markov chain model, fitting functions that represent distributions of stage durations can be compared through the parameter values if they are from the same family of functions, for example the Weibull density function family. Interpreting a state bout duration as a "time-to-failure" as in statistical reliability theory, the Weibull distribution indicates a distribution where the failure rate is proportional to a power of time. As shown in Figure 3.3, the shape parameter k determines one of several cases: if $k < 1$, then failure rate decreases over time; if $k = 1$, then it does not change over time; otherwise failure rate increases with time.

3.3.2 Markov Chain Evaluation

A Markov chain provides a simple dynamical model of the transitions between sleep stages in Table 3.1. Noticeably, the self-transition probabilities are much higher than the inter-state transition probabilities.

Table 3.1: Stage transition matrix for Markov chain model.

Sleep Stage	**Wake**	**NREM**	**REM**
Wake	0.9207	0.0764	0.0029
NREM	0.0239	0.9705	0.0056
REM	0.0216	0.0108	0.9676

As in section 3.3.1, the equilibrium distribution of the Markov chain model is obtained by normalizing the eigenvalue 1 in the eigenvector of the Markov transition matrix. The equilibrium distribution is given in Table 3.2. It characterizes the long-term dynamics of human sleep as captured by this Markov chain model.

Table 3.2: Equilibrium distribution for Markov chain model

Equilibrium	**Wake**	**NREM**	**REM**
Value	0.23	0.64	0.13

3.3.3 Semi-Markov Chain Evaluation

Semi-Markov models have similar inter-stage transition behavior to Markov models. But in contrast with the Markov chain model, the semi-Markov chain model allows an explicit representation of the stage duration distributions. Training a Markov chain model requires only the calculation of the probability transition matrix. Building hidden Markov model needs extra computation of the emission matrix. Semi-Markov chains are required to estimate state bout duration distributions.

The stage transition matrix obtained for the semi-Markov chain model is given in Table 3.3. Note that the self-transition probabilities are 0. This matrix can be calculated from the Markov chain model transition probability matrix by normalizing the inter-state probabilities in each row. This very simple operation produces values that more directly reveal the relative likelihoods of various inter-stage transitions.

Table 3.3: Stage transition matrix for semi-Markov chain model.

Sleep Stage	Wake	NREM	REM
Wake	0	0.9632	0.0368
NREM	0.8093	0	0.1907
REM	0.6655	0.3345	0

For stage bout durations, power law, and Weibull distribution functions [15] are considered as candidates for sleep stages. The specifications in distribution functions are listed as follows:

1. For exponential function, the expected value of the exponential distribution for a given sleep stage in MLE corresponds to the expected value of durations over the sleep stage data. The maximum length of stage bout duration for all sleep stages is 230 epochs. Therefore, the initial values for μ can be taken to be between 1 and 230.

2. For power law function, x_{min} should be 1, and a large enough range for initial values for α is the range from 1 to 10 times the MLE estimate of α. A delta increment value of 0.01 is used to select sufficient α values in this range. n is 244 patients.

3. For Weibull function, to be consistent with the kernel estimated data distribution, k is limited to a range between 0 and 2. λ can range from 1 to 230 (the maximum duration of any sleep stage).

We apply the search procedure and decide the best fitting parameters as described in section 3.2.1. The experimental results on three versions of datasets are shown as follows:

- *WNR Results.*
  The Weibull family of distributions was selected to model the REM stage durations in the semi-Markov model illustrated in Figure 3.6 , for example. The results are summarized in Table 3.4. It can be seen that for wake stage, the best fitting function is determined by least squares error estimation. For NREM and REM, the best fitting functions are determined by maximum likelihood estimation. Table 3.5 provides the parameter values for the best Weibull function fits found. The fact that the Weibull shape parameters for wake, NREM, REM are, respectively, smaller than, approximately equal to, and larger than 1, show that stage bout durations for these three stages have characteristic and distinct behaviors. Specifically, the "failure rate" for wake, NREM, and REM decreases, does not change, and increases over time, respectively. These very different behaviors, which are modeled precisely in the semi-Markov model, cannot be captured at all by the Markov chain model in section 3.3.2, for which all stage bout durations are exponentially distributed (constant failure rate, in the reliability metaphor). The results are in contrast with reports by other researchers [11] that the durations of wake stage, NREM sleep stage, and REM sleep stage follow power laws or single exponential distributions.

Table 3.4: Goodness of fit (GOF) values, as defined in section 3.2.1. The lower this value, the better the fit. The best fit(s) for each stage is(are) underlined. EDF, PDF, and WDF denote exponential, power law, and Weibull density functions, respectively. ML and LS denote the search approaches described in section 3.2.1.

ML	**Wake**	**NREM**	**REM**
EDF	6.2963	<u>0.5543</u>	1.7284
PDF	2.0704	7.6185	6.0023
WDF	1.0418	<u>0.5684</u>	<u>0.3046</u>
LS	**Wake**	**NREM**	**REM**
EDF	5.0303	0.6352	2.0428
PDF	2.0704	5.8969	11.8552
WDF	<u>0.7788</u>	0.8949	<u>0.3198</u>

Table 3.5: Parameter values for the Weibull function fits.

Parameters	**Wake**	**NREM**	**REM**
λ **(scale)**	4.024	33.74	33.35
k **(shape)**	0.4378	1.000	1.286

- *WDL Results.*

  In Figure 3.7, stage bout durations at deep and light stages do not follow linear distribution under logarithmic scale in x and y axes. Thus power law does not fit well stage bout data visually in these two stages. Furthermore, we find that for wake stage, the best fitting function is determined by least squares error estimation. For deep and light sleep stages, the best fitting functions are also derived from least squares error estimation. The fact that the Weibull approximates better than exponential at the starting section between 0 and 2 of log stage duration leads to superiority of Weibull curve fitting results. In sum, scale and shape parameters allows Weibull to be a smoothing slope, which fits stage bout durations well.

- *W5 Results.*

  Similar to data distribution in Figure 3.8, stage bout duration at stage 1 is consistent with linear one under logarithmic axes and power law perform best than other two fitting results. For stage 1, the fitting function is selected by least squares error estimation. For stage 2, the best fitting function is derived from maximum likelihood estimation. Exponential fit competes well with the Weibull one in stage 2 since bout duration shape accounts for exponential estimation. In addition, simulation of exponential properties make Weibull distribution perform as well as exponential one.

Figure 3.9 depicts for each sleep stage a comparison between the exponential fit of the sleep stage duration distribution used by the Markov chain model, and the Weibull fit used in the semi-Markov chain model. It can be seen that the Weibull functions used in the semi-Markov chain model provide a much better fit for the wake and REM stages than the exponential functions used by the Markov chain model do. As examples, the exponential Markov fit underestimates the probability of longer wake bouts, and overestimates the probability of shorter REM bouts. The Weibull semi-Markov fit captures both of these behaviors well, by adjusting the shape parameter appropriately. The Markov and semi-Markov fits coincide in the case of NREM, as NREM durations are relatively well modeled by an exponential distribution.

The semi-Markov chain model's empirically observed equilibrium distribution is given in Table 3.6. It provides the asymptotic probabilities of stages over a collection of simulation runs where the length of simulations ranged from 1,000 to 1,000,000. The semi-Markov chain equilibrium distribution in Table 3.6 is observed to be very close to that of the Markov chain model shown in Table 3.2.

Table 3.6: Equilibrium distribution for semi-Markov chain model

Equilibrium	Wake	NREM	REM
Limit	0.25	0.61	0.14

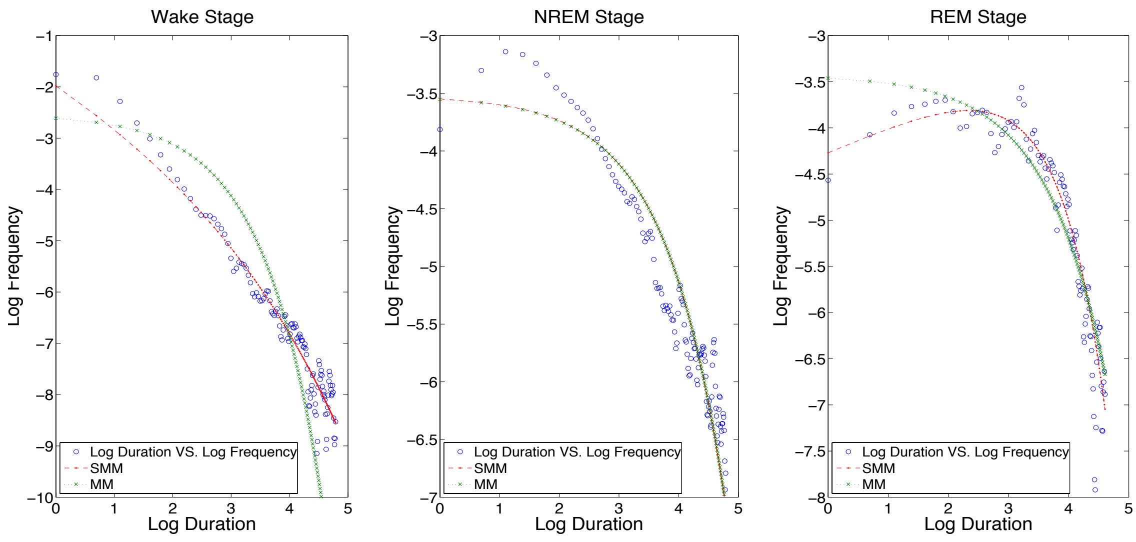


Figure 3.9: Comparison of fits for the stage duration distribution function used by the Markov chain model (MM), exponential, and by the semi-Markov chain model (SMM), Weibull.

Chapter 4

Convergence of the CDMC Framework

CDMC is a general algorithmic strategy for simultaneous clustering and dynamical modeling of sequence data [34]. CDMC's pseudocode is shown in Algorithm 1. Similar to generalizing the method in [33], it is model-based clustering using dynamical state-based models like Markov models, including Markov chain, semi-Markov chain, and hidden Markov models. However, [33] utilized clustering data to estimate specific parameters for every instance, which is subject to high variance in small gene expression data. [31] provided abundant temporal information for each web navigation sequences in practice. Furthermore, [81] proposed a similar unifying probabilistic framework, clustering individuals into groups in terms of observed relevant objects. From a different perspective of model-based clustering, [82] presented a bipartite graph, in which it considered instances with generative models as generalized cluster centroids, by the way of taking the generative likelihood information as a proximity measure.

The simultaneous modeling and clustering strategy is closely related to the more general Expectation-Maximization (*EM*) framework [83]. The *EM* training process alternates between expectation step and maximization step, which possesses functional similarity as what the CDMC algorithm does. Furthermore, an iterative procedure of training models on a given dataset has been applied in data mining and machine learning techniques. It covers classifications (e.g., neural network and Markov models) and clustering (K-means clustering) methods. A key point is concerned with convergence property, especially for training models in a non-closed functional form. Mathematical proof gives its support of iteratively training over data to calculate a solution gradually. For example, it has been proved that the convergence of *EM* framework to local extremes (See Figure 4.1) is subject to variance of actual data distribution, during the iterative process of expectation step and maximization step [83, 84]. In other words, *EM* algorithm is heavily dependent on domain knowledge [85], in particular on time series, such as human sleep and web user navigation datasets. Likewise, the convergent problem in the CDMC framework is proved to be inherited from *EM* framework.

In this chapter, we review *EM* algorithm in section 4.1, present basic proof of convergence property on iterative computation of maximum-likelihood in CDMC in section 4.2, the convergence speed brought up by stopping criteria in section 4.3, as well as experimental results and analysis on convergence using synthetic dataset in section 4.4.

4.1 Expectation-Maximization Algorithm

We review the general definitions of the Expectation (*E*) and Maximization (*M*) steps in the *EM* algorithm [83]. To provide clear descriptions of the *EM* algorithm, unambiguous mathematical notations are defined as follows:

- Let $X = \{x_1, \ldots, x_n\}$ denote the observable part of the full data Y, which consists of n independently data obtained from an underlying probability distribution [86]. The underlying probability distribution is described by a set of parameters θ yet to be determined.

- Let $Z = \{z_1, \ldots, z_n\}$ represent the unobservable part of Y. Y consists of X and Z, i.e. $Y = X \cup Z$. Note that the unobserved Z is a latent variable whose probability distribution depends on the unknown parameters θ and the observed data X. Likewise, Y is considered as a random variable due to the existence of Z in Y.

- $\theta^{(t)}$ refers to as the estimated values of θ in the current (e.g. t^{th}) iteration and $\theta^{(t+1)}$ as the re-estimated values of θ on the next (e.g. $(t+1)^{th}$) iteration of the *EM* algorithm in terms of $\theta^{(t)}$ [87].

The *EM* algorithm aims at finding the maximum likelihood estimation of θ so as to maximize $E[\log P(Y|\theta)]$ [86]. The expectation of the log-likelihood of $P(Y|\theta)$, $E[\log P(Y|\theta)]$, is taken over the probability distribution with respect to the full data Y given the unknown parameters θ. Since the latent variable $Z \subset Y$ is unknown, the *EM* algorithm actually repeatedly re-estimates the expected value of Z (i.e., $E(Z)$) given its current hypothesis $\theta^{(t)}$ and re-calculates the maximum likelihood hypothesis $\theta^{(t+1)}$ using $E(Z)$ instead of Z.

Consider $E[\log P(Y|\theta)]$ as a function of θ denoted by $Q(\theta^{(t+1)}|\theta^{(t)})$, which indicates the iterative process of evaluating the maximum likelihood hypothesis θ. In general, the *EM* algorithm repeats two steps of Estimation (E) and Maximization (M) steps until convergence [86]:

- Estimation (E) step: Calculate $Q(\theta^{(t+1)}|\theta^{(t)})$ using the current hypothesis $\theta^{(t)}$ and the observable data X to estimate the hidden probability distribution over Y.

$$Q(\theta^{(t+1)}|\theta^{(t)}) \leftarrow E[\log P(Y|\theta^{(t+1)})|\theta^{(t)}, X]$$

- Maximization (M) step: Substitute the hypothesis $\theta^{(t+1)}$ that maximizes this $Q(\theta^{(t+1)}|\theta^{(t)})$ function for the hypothesis $\theta^{(t)}$.

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta^{(t+1)}} Q(\theta^{(t+1)}|\theta^{(t)})$$

Note that in the estimation step, actually Z is replaced by the expectation of Z (i.e., $E(Z)$). In the context of considering $P(Y|\theta)$ as mixture models (see section 4.2), one can determine the probability of each possible value (class label) of Z for each data x_i $(i = 1 \dots n)$ and then it is called soft *EM* algorithm. Otherwise, it is called hard *EM* algorithm when making hard choice of Z, each z_i with 0 or 1. The hard *EM* algorithm is a procedure listed as follows:

1. Initialize $\theta^{(t)}$ (it is often application specific).

2. Repeat the following two steps until convergence:

   (a) $\{z_1, \dots, z_n\} \leftarrow \arg\max_{\{z_1,\dots,z_n\}} \{M_{\{z_1,\dots,z_n\}}|\theta^{(t)}, X\}$

   (b) $\theta^{(t+1)} \leftarrow \arg\max_{\theta^{(t+1)}} \{E[\log M(Y|\theta^{(t+1)})|\theta^{(t)}, X, z_1, \dots, z_n]\}$

In sum, the hard *EM* algorithm can be interpreted as that when θ is known, one can find the value of latent variable Z by maximizing the log-likelihood over all possible values of Z. On the other hand, if Z is predetermined, one could find an estimate of the parameters θ by clustering the observed data X according to previously known Z by maximum-likelihood estimation.

4.2 Convergence Properties of CDMC Framework

Collective dynamical modeling & clustering framework in Algorithm 1 simultaneous groups and dynamically builds models over sequence data. We prove that CDMC follows *E-M-M* approach, that is *EM* algorithm with an additional *M* step of hard cluster assignment.

Before the proof of convergence in CDMC, it is necessary to provide clear descriptions of the CDMC algorithm, and unambiguous mathematical notations are defined as follows:

- Let Y represents the full data, including the known time series distribution (D) and the unknown cluster membership labels (C).
- Let $D = \{x_1, \dots, x_n\}$ denote the observable part of the full data Y, which consists of n independently data obtained from an underlying probability distribution. The underlying probability distribution is described by a set of parameters θ yet to be determined.
- Let $C = \{c(x_1), \dots, c(x_n)\}$ represent the unobservable part of Y. Y consists of D and C, i.e. $Y = D \cup C$. Note that the unobserved C is a latent variable whose probability distribution depends on the unknown parameters θ and the observed data D. Likewise, Y depends on the parameters θ due to the presence of C in Y.
- Assuming that there are k different clusters, denote the corresponding distributions of the full data in these clusters by $M(Y|\theta) = \{M_1(Y|\theta_1), \dots, M_k(Y|\theta_k)\}$, where $\theta = \{\theta_1, \dots, \theta_k\}$ is a set of unknown parameters.
- $\theta^{(t)}$ refers to as the estimated values of θ in the current t^{th} iteration of the model estimation step in Algorithm 1 and $\theta^{(t+1)}$ as the re-estimated values of θ on the next $(t+1)^{th}$ iteration of the CDMC algorithm in terms of $\theta^{(t)}$.

The CDMC algorithm aims at finding the maximum likelihood estimation of θ so as to maximize $E[\log M(Y|\theta)]$ while stabilizing clustering C. The expectation of the log-likelihood of $M(Y|\theta)$, $E[\log M(Y|\theta)]$, is taken over the probability distribution with respect to the full data Y given the unknown parameters θ. Consider $E[\log M(Y|\theta)]$ as a function of θ where all pairs $(\theta^{(t+1)}, \theta^{(t)})$ exist and let

$$Q(\theta^{(t+1)}|\theta^{(t)}) = E[\log M_{c(x)^{(t+1)}}(Y|\theta^{(t+1)})|\theta^{(t)}, x] \tag{4.1}$$

that indicates the iterative process of evaluating the maximum likelihood hypothesis θ. Since the latent variable $C \subset Y$ is unknown, the CDMC algorithm actually repeatedly re-estimates C and calculate $Q(\theta^{(t+1)}|\theta^{(t)})$ given its current hypothesis $\theta^{(t)}$ and the observable part x in D, and re-calculates the maximum likelihood hypothesis $\theta^{(t+1)}$ using the current estimates of C.

In general, the CDMC algorithm repeats two steps of Cluster Assignment (*CA*) and Model Estimation (*ME*) steps until convergence of clusters in Algorithm 1:

- Cluster Assignment (*CA*) step: Assign each instance x in D to the cluster $c(x)^{(t+1)}$ having the model $M_{c(x)_{(t+1)}}(Y|\theta)$ most likely to generate x and compute $Q(\theta^{(t+1)}|\theta^{(t)})$ given its current hypothesis $\theta^{(t)}$, the observable data D, and updated clustering assignment $c(x)^{(t+1)}$ to estimate the hidden probability distribution over Y.

$$\begin{aligned} c(x)^{(t+1)} &\leftarrow \arg \max_{c(x)^{(t+1)}} \{M_{c(x)^{(t+1)}}(Y|\theta^{(t)})|\theta^{(t)}\} \\ Q(\theta^{(t+1)}|\theta^{(t)}) &\leftarrow E[\log M_{c(x)^{(t+1)}}(Y|\theta^{(t+1)})|\theta^{(t)}, x] \end{aligned} \tag{4.2}$$

- Model Estimation (*ME*) step: Train each dynamical model M_i with maximum data likelihood for each cluster c_i $\{i = 1, \ldots, k\}$ by calculating $Q(\theta^{(t+1)}|\theta^{(t)})$ using the current hypothesis $\theta^{(t)}$, and then substituting the hypothesis $\theta^{(t+1)}$ that maximizes this Q function for the hypothesis $\theta^{(t)}$.

$$\theta^{(t+1)} \leftarrow \arg \max_{\theta^{(t+1)}} \{Q(\theta^{(t+1)}|\theta^{(t)})\} \tag{4.3}$$

Note that the cluster assignment step involves not only the E step in the EM algorithm, which calculates the expectation of the log-likelihood of $M(Y|\theta)$, $E[\log M(Y|\theta)]$, but also the additional M step of hard assignment to the most likely cluster C. The model estimation step just includes M step in the EM algorithm. In sum, the above model estimation and clustering assignment steps correspond to E and M steps in the EM algorithm (see section 4.2) with the additional M step of hard assignment, except convergence conditions: convergence of clusters in CDMC algorithm and convergence of maximum likelihood in EM algorithm.

Consider a mixture of dynamical models using the above procedure in CDMC as an example. A dataset $D = \{x_1, \ldots, x_n\}$ is generated by a mixture of k distinct distributions where data points are mapped along the x axis. The goal is to simultaneously output $\theta = \{\theta_1, \ldots, \theta_k\}$ that describes the mixed probability distributions $M(\theta)$ and clustering results $C = \{c(x_1), \ldots, c(x_n)\}$. Among $M(\theta) = \{M_1(\theta_1), \ldots, M_k(\theta_k)\}$, the probability of M_i $\{i = 1, \ldots, k\}$ generating a single datum $y_i = \langle x_i, c(x_i)\rangle$ in the full data Y can be written as

$$M(y_i|\theta^{(t+1)}) = M(x_i, c(x_i)|\theta^{(t+1)}) = \sum_{j=1}^{k} \mathbb{I}(c(x_i) = j) M_{c(x_i)}(\theta_{c(x_i)}^{(t+1)}) \tag{4.4}$$

where $\mathbb{I}(c(x_i) = j)$ is an indicator function that it is 1 only if x_i belongs to the j^{th} probability distribution and 0 otherwise, in accordance with constraints in hard EM algorithm. From another perspective, the indicator function gives the prior distribution of x_i generated by the j^{th} hidden probabilistic distribution.

Furthermore, the expression for $M(Y|\theta^{(t+1)})$ is

$$M(Y|\theta^{(t+1)}) = \prod_{i=1}^{n} M(y_i)|\theta^{(t+1)}) \tag{4.5}$$

Given the probability $M(y_i|\theta^{(t+1)})$ for a single instance y_i, the logarithmic function of the probability $\log M(Y|\theta^{(t+1)})$ for all n instances in the dataset D is

$$\begin{aligned}\log M(Y|\theta^{(t+1)}) &= \log\prod_{i=1}^{n} M(y_i|\theta^{(t+1)}) = \sum_{i=1}^{n}\log\ M(y_i|\theta^{(t+1)}) \\ &= \sum_{i=1}^{n}\left[\log\sum_{j=1}^{k}\mathbb{I}(c(x_i)=j)M_{c(x_i)}(\theta_{c(x_i)}^{(t+1)})\right]\end{aligned} \tag{4.6}$$

Calculate the expected value of this $\log M(Y|\theta^{(t+1)})$ over the distribution of unobserved part C of Y. That is,

$$\begin{aligned}E\left[\log M(Y|\theta^{(t+1)})\right] &= E\left[\sum_{i=1}^{n}\sum_{j=1}^{k}\mathbb{I}(c(x_i)=j)\left(\log^{M_{c(x_i)}}+\log^{\theta_{c(x_i)}^{(t+1)}}\right)\right] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k}E\left[\mathbb{I}(c(x_i)=j)\right]\left[\left(\log^{M_{c(x_i)}}+\log^{\theta_{c(x_i)}^{(t+1)}}\right)\right]\end{aligned} \tag{4.7}$$

Note that since $\log M(Y|\theta^{(t+1)})$ is a linear function of those $c(x_i)\in C$ $(i=1,\ldots,n)$ and the linearity of the expectation (e.g.$E[f(c)]=f(E[c])$), the above equations hold. The expected value $E[\mathbb{I}(c(x_i)=j)]$ of each latent variable $c(x_i)$ is 1 if x_i belongs to the j^{th} distribution, assuming the current hypothesis $\theta^{(t)}=\{\theta_1^{(t)},\ldots,\theta_k^{(t)}\}$ holds, and 0 otherwise.

Consider X as "incomplete data" and Y as "complete data". It implies the existence of two sample spaces (e.g., $\mathscr{Y}$ and $\mathscr{X}$) and a many-one mapping from $\mathscr{Y}$ to $\mathscr{X}$ represented as [83]:

$$\begin{aligned}g(\mathbf{x}\mid\theta) &= \int_{\mathscr{Y}(\mathbf{x})} f(\mathbf{y}\mid\theta)dy \\ Q(\theta^{(\mathbf{t+1})}\mid\theta^{(\mathbf{t})}) &= E(\log f(\mathbf{y}\mid\theta^{(\mathbf{t+1})})\mid\mathbf{x},\theta^{(\mathbf{t})})\end{aligned} \tag{4.8}$$

For CDMC, $\log M(Y|\theta^{(t+1)})$ here refers to $\log f(\mathbf{y}\mid\theta^{(\mathbf{t+1})})$ under the general *EM* algorithm, for which [83] proved that the likelihood sequence of the general *EM* algorithm is non-decreasing and convergent. Additionally, [84] corrected the use of the triangle inequality and pointed out two useful special cases:

- If Y is denoted as a curved exponential family with compact parameter space, then the limit points of any *EM* sequences are stationary points of the log-likelihood function $\log M(Y|\theta^{(t+1)})$.
- If the log-likelihood function $\log M(Y|\theta^{(t+1)})$ is unimodal with several satisfied differentiability condition, then any *EM* sequences converges to the unique maximum likelihood estimate $\theta^{(t+1)}$.

In collective dynamical modeling & clustering framework, state-based dynamical models such as Markov models are taken and state duration distributions are of exponential, power law, and Weibull density functions that follow curved exponential formulations. In addition, all of those probability density functions are differential among ranges of all values. Therefore, the limit points of an $Q(\theta^{\mathbf{(t+1)}} \mid \theta^{\mathbf{t}})$ sequences in CDMC are stationary points of the log-likelihood function $\log M(Y|\theta^{(t+1)})$ and the $Q(\theta^{\mathbf{(t+1)}} \mid \theta^{\mathbf{t}})$ sequences in CDMC converge to maximum likelihood estimate as hard clustering agreement sequences converge in the same trend as *EM* sequences do.

Finally, [88] show that under general, simple, and verifiable conditions, any *EM* sequence is convergent. In a conclusion, under the assumption of continuity of the function $Q(\theta^{\mathbf{(t+1)}} \mid \theta^{\mathbf{t}})$ where all pairs $(\theta^{(t+1)}, \theta^{t})$ exist, CDMC framework converges to limit points over the entire parameter space if clustering agreement sequences in CDMC behaves similar convergent trend as *EM* sequences.

4.3 Stopping Criteria

Stopping criteria are used as a measure of agreement between clusterings in Algorithm 1. We apply Rand index, adjusted Rand index, and normalized mutual information to have a measure for the similarity for a pair of clusterings [89].

Let n be the number of sequences in time series dataset D. $\mathbb{C} = \{C_1, \dots, C_k\}$ and $\bar{\mathbb{C}} = \{\bar{C}_1, \dots, \bar{C}_l\}$ are two clustering results of D. n_{11} is the number of pairs of sequences that are in the same cluster under $\mathbb{C}$ and $\bar{\mathbb{C}}$. n_{00} is the number of pairs of sequences that are in different clusters under $\mathbb{C}$ and $\bar{\mathbb{C}}$. n_{10} is the number of pairs of sequences that are in the same cluster under $\mathbb{C}$ but not in $\bar{\mathbb{C}}$. n_{01} is the number of pairs of sequences that are in different clusters under $\mathbb{C}$ but the same in $\bar{\mathbb{C}}$. A confusion matrix CM of the pair $\mathbb{C}$ and $\bar{\mathbb{C}}$ consists of $k \times l$ entries in the matrix. For example, an entry in the i-th row and the j-the column is $m_{ij} = |C_i \cap \bar{C}_j|, 1 \leq i \leq k, 1 \leq j \leq l$, which denotes the number of sequences in both clusters C_i and $\bar{C}_j$.

4.3.1 Rand Index

Rand index (RI) [90] interprets clustering as a set of decisions for totally $n(n-1)/2$ pairs of sequences in D. The Rand index assigns two similar sequences to the same clusters (true positive), and penalizes false positive and false negative during clustering. It is defined as:

$$\mathbb{R}(\mathbb{C}, \bar{\mathbb{C}}) = \frac{2(n_{11} + n_{00})}{n(n-1)} \tag{4.9}$$

R ranges from 0 to 1. It is proved that Rand index is highly dependent on the number of clusters [91]. Furthermore, it produces a nonzero value for the comparison of two randomly constructed partitions.

4.3.2 Adjusted Rand Index

Adjusted Rand index (ARI) [92] is based on the Rand index but corrects for clustering agreements due to chance. The adjusted indices takes a generalized hypergeometric distribution as null hypothesis, in which the two clusterings $\mathbb{C}$ and $\bar{\mathbb{C}}$ are drawn randomly with a predefined number of clusters and a fixed number of elements in each cluster. It is defined as the difference of $\mathbb{R}(\mathbb{C},\bar{\mathbb{C}})$ and its expected value:

$$\mathbb{R}_{adj}(\mathbb{C},\bar{\mathbb{C}}) = \frac{\sum_{k}^{i=1}\sum_{l}^{j=1}\begin{pmatrix} m_{ij} \\ 2 \end{pmatrix} - t_3}{\frac{1}{2}(t_1+t_2)-t_3} \tag{4.10}$$

where $t_1 = \sum_{i=1}^{k}\begin{pmatrix} |C_i| \\ 2 \end{pmatrix}$, $t_2 = \sum_{j=1}^{l}\begin{pmatrix} |\bar{C}_j| \\ 2 \end{pmatrix}$, and $t_3 = \frac{2t_1t_2}{n(n-1)}$. Note that due to the uncertainty of number of elements in clusters under the null hypothesis, $\mathbb{R}_{adj}(\mathbb{C},\bar{\mathbb{C}})$ produces negative index values [93].

4.3.3 Normalized Mutual Information

Normalized mutual information (NMI) [94] is a distinct way of evaluating clusters by the tradeoff between the number of clusters and qualities. It satisfies three properties [95]: (1) Consistency with the set of clusterings; (2) Robustness to small variations in the set of clusterings; (3) Goodness of fit with the ground truth information. The normalized mutual information between clusterings $\mathbb{C}$ and $\bar{\mathbb{C}}$ is defined as:

$$\mathbb{NMI}(\mathbb{C},\bar{\mathbb{C}}) = \frac{2\mathbb{I}(\mathbb{C},\bar{\mathbb{C}})}{\mathbb{H}(\mathbb{C})+\mathbb{H}(\bar{\mathbb{C}})} \tag{4.11}$$

where the mutual information between $\mathbb{C}$ and $\bar{\mathbb{C}}$ is $\mathbb{I}(\mathbb{C},\bar{\mathbb{C}}) = \sum_{i=1}^{k}\sum_{j=1}^{l} P(i,j)\log_2\frac{P(i,j)}{P(i)P(j)}$. $\mathbb{H}(\mathbb{C}) = -\sum_{i=1}^{k} P(i)\log_2 P(i)$ is the entropy [96] associated with clustering $\mathbb{C}$. $P(i) = \frac{|C_i|}{n}$ and $P(i,j) = \frac{|C_i \cap \bar{C}_j|}{n}$. The range of $\mathbb{NMI}(\mathbb{C},\bar{\mathbb{C}})$ is between 0 and 1.

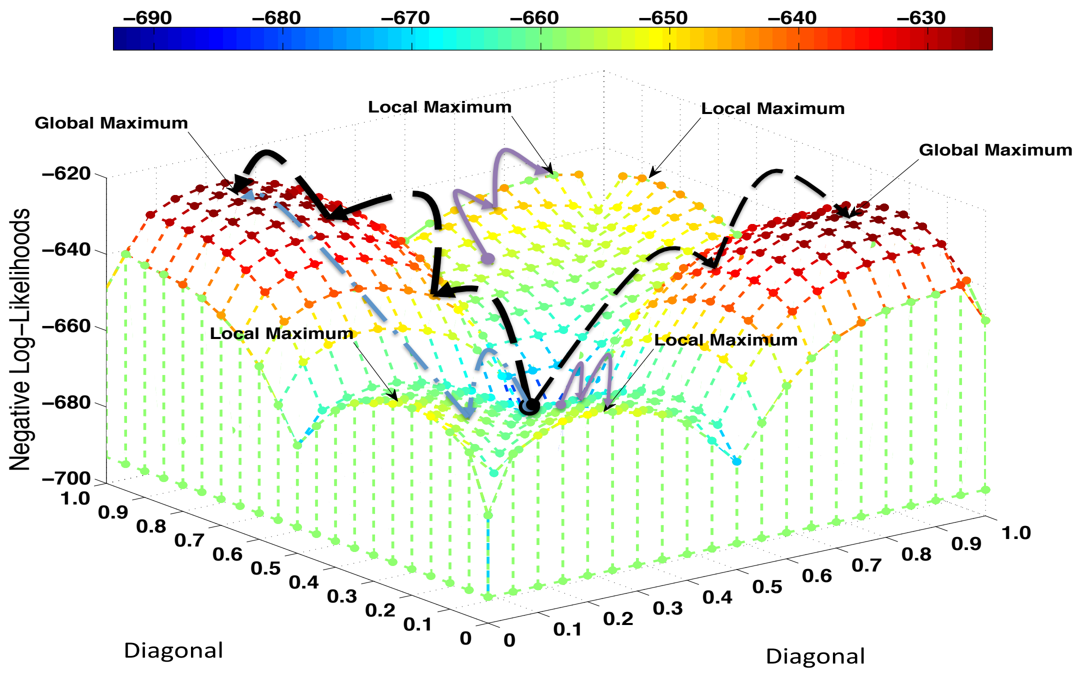


Figure 4.1: Local and global extremes across parametric space. A mixture Markov data comes from two hidden Markov models (HMMs), each with two states. Values along the main diagonals (named "diagonal" here) in the transition matrix of a given HMM are the same. Given each setup of the two HMM over the entire parameter space, the negative log-likelihoods are calculated. There are two global maxima and four local maxima extremes symmetrically distributed in the space. Under different initializations of state transitions in HMM, CDMC could make training the two HMMs converge to global maxima and local maxima.

4.4 Experimental Evaluation

The goal of this experimental study is to investigate the effect of stopping criteria in convergence for CDMC framework. We will show that Rand index is the best choice for stopping criterion in CDMC framework in terms of (i) number of iterations to converge and (ii) accuracy of classification in CDMC.

4.4.1 Experimental Setup & Methodology

In experimental study, we compare three index metrics in section 4.3 as the basis for the stopping criterion in CDMC (Algorithm 1), using the resulting CDMC convergence time (number of modeling-clustering iterations) and classification accuracy over labeled synthetic data for evaluation. The prerequisite setups include:

- **Stopping Criteria.**

  Standard Rand index (RI), adjusted (chance-corrected) Rand index (ARI), and Normalized Mutual Information (NMI), were used respectively in CDMC, and the results were compared in terms of the resulting classification accuracy and number of iterations to convergence.

- **Synthetic semi-Markov Mixture Dataset.**

  A synthetic semi-Markov mixture dataset was used in the evaluation of stopping metrics. It is generated from two distinct semi-Markov chains, each with two states, but with different transition probability matrices and state duration statistics. In the experiments, $N = 100$ sequences, each of length $L = 100$, were used in all trials. The following transition matrices and Weibull (shape, scale) parameter values (in that order) were used for the two semi-Markov chains. There were two Weibull distributions per chain, one for each state.

$$\textit{Semi-Markov chain 1:} \quad \begin{pmatrix} 0.90 & 0.10 \\ 0.10 & 0.90 \end{pmatrix}, \quad (3,5), \ (2.5, 4,5)$$

$$\textit{Semi-Markov chain 2:} \quad \begin{pmatrix} 0.15 & 0.85 \\ 0.85 & 0.15 \end{pmatrix}, \quad (3,4), \ (2.5, 3,5)$$

  This allows the use of classification accuracy to evaluate performance of clusterings.

4.4.2 Convergence For Stopping Criteria

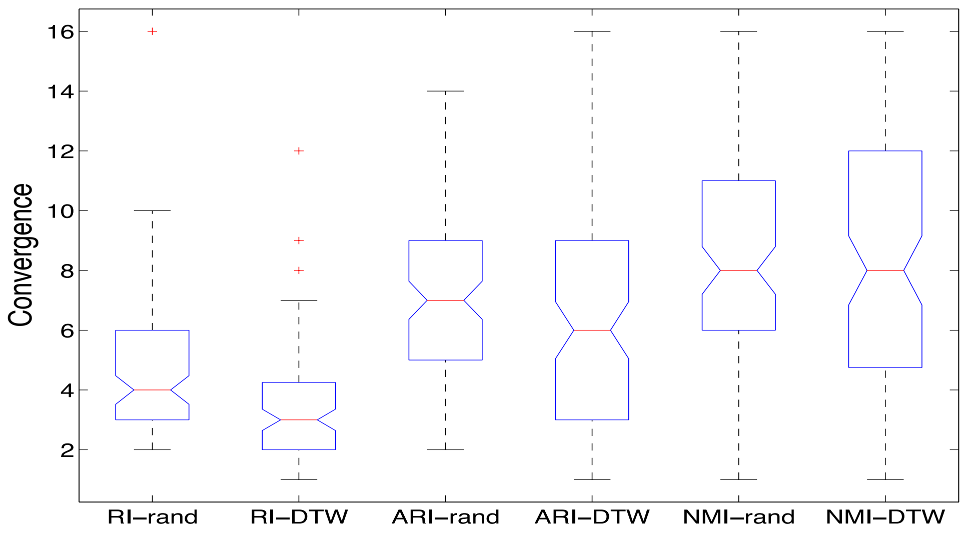


Figure 4.2: Iterations to convergence for different stopping criteria. Random and DTW-initialized CDMC. Semi-Markov mixture data. Stopping metrics: RI (two left), ARI (two center), NMI (two right). Non-overlapping notches indicate significant difference in medians ($p < 0.05$) using a Wilcoxon rank sum test (Mann-Whitney test). RI stopping is significantly faster than others. DTW initialization significantly speeds up convergence for RI stopping.

The criterion for stopping the iterative process in CDMC depends on the clustering similarity metric. Three similarity metrics were compared (see section 4.3): the standard Rand index (RI), the adjusted Rand index (ARI) intended to correct for chance clustering agreements, and Normalized Mutual Information (NMI). The resulting numbers of iterations required for convergence are shown in Figure 4.2. RI is seen to lead to a significantly lower median number of iterations to convergence (4 and 3, for pseudorandom initialization and DTW initialization, respectively) as compared with ARI (7 and 6 iterations) and NMI (8 iterations for both random and DTW initialization). DTW initialization provides significantly faster convergence than pseudorandom initialization in the case of the Rand index as the similarity metric (left two boxes in Fig. 4.2).

The faster convergence observed for RI as compared with ARI is in itself not surprising, as the numerical threshold *minSim* used for stopping in Algorithm 1 is the same for the three similarity metrics, while ARI has lower values than RI due to the intended correction of clustering agreement due to chance. One would therefore also expect that the more stringent ARI criterion would lead to better differentiated clusters. However, as Figure 4.3 shows, the resulting median accuracy values from testing over labeled synthetic semi-Markov mixture data are not significantly different at the level $p < 0.05$ (Wilcoxon-Mann-Whitney test). The combination of faster convergence and comparable accuracy points to RI as the superior choice of stopping criterion for CDMC.

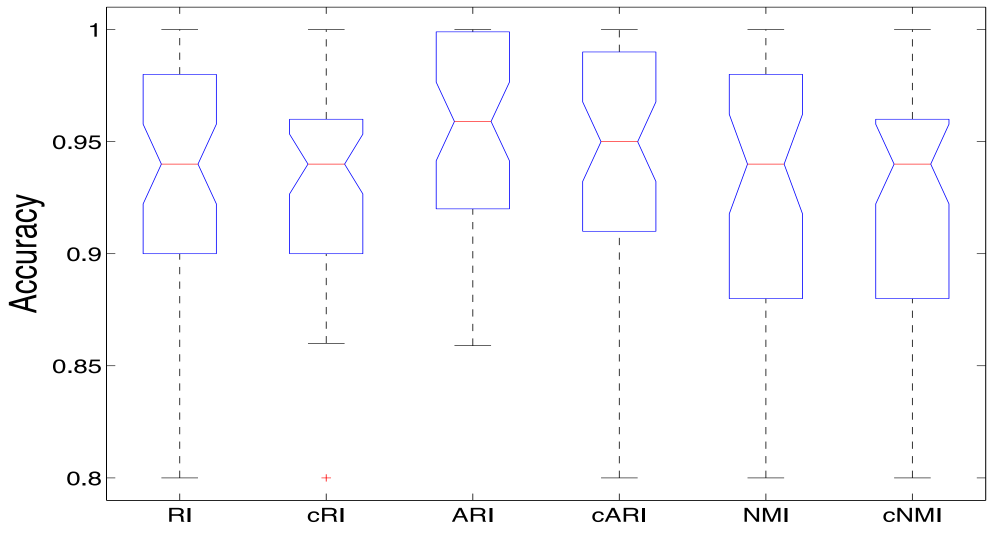


Figure 4.3: Accuracies for different stopping criteria of DTW-initialized and constrained DTW-initialized CDMC over semi-Markov mixture data. Stopping metrics: RI (two left), ARI (two center), NMI (two right). Non-overlapping notches indicate significant difference in medians ($p < 0.05$). Median accuracy values do not differ significantly.

In section 4.3, adjusted Rand index defines the CDMC stopping criterion in order to correct for clustering agreements due to chance. Nevertheless, ARI as the clustering similarity metric does not lead to significantly more accurate clusterings, while it does significantly increase the number of iterations required for convergence as compared with the standard Rand index. Similar statements hold for the Normalized Mutual Information metric as compared with the standard Rand index. Therefore, the standard Rand index is the best choice of similarity metric in this context among these candidates.

Chapter 5

Distributed Deployment Architecture

Terabytes and even petabytes of data are becoming the norm in organizations today. However, traditional data mining techniques do not accommodate such large-volume, complex data in memory, nor do they have the ability to train models fast on this large scale data [97]. In order to address the challenges of big data analytics in science and engineering domains [98], novel ways of processing large scale data [99] and new architectures and infrastructures [100] need to be proposed. It is time to construct new frameworks supporting more powerful computations and more comprehensive analysis in the face of the exponential data growth.

For collective dynamical clustering modeling framework, it is originally designed to hold all useful data in memory do clustering and modeling steps iteratively [34], since processors get access to data faster than in external disks. However, in the context of big data, it is often impossible to achieve this goal and thus disk I/O becomes a bottleneck in system performance. Even worse is when a fast response time is with the highest priority [101]. Additionally, each input instance is represented as a collection of features in the framework. These features produced during modeling step take up extra memory. If features grows dramatically in memory, the process of extracting features runs out of memory and stores additional results in disks. This exacerbates disk I/O performance. Lastly, in this framework, it is an iterative process of running clustering and modeling steps until it reaches some stable status. Each iteration repeats extracting features given the same input data, which takes up a big portion of CPU time and slows down modeling data in a sequential mode. The iteration even increase the number of disk I/O and thus potentially increase the response time of the system. In sum, it is urgent to make this framework capable of dealing with large-scale datasets.

One way of processing large-scale datasets of discrete time series is to apply dimensionality reduction for compressing original data so as to load more data instances into memory. As an important part of data pre-processing for machine learning, dimensionality reduction is to identify a reduced collection of data features that represents the original dataset with minimal information loss. Depending on the data type of sequential data, there are mainly two kinds of dimensionality reduction techniques. One is for sequences of nominal (or symbolic) values. The other one is for sequences of numeric values. For symbolic sequences, dimensionality reduction could be further partitioned into two primary features: global feature [102] and local feature (i.e., sequence element feature [103], k-grams feature [104], k-gapped feature [105], and pattern-based feature [106, 107, 108]). For numeric sequences, a discretization process may be utilized to change numeric sequences into symbolic sequences. Then, those traditional ways of dimensionality reduction such as feature selection and extraction techniques could be applied. However, it may lead to the problem of information loss. Researcher developed specific techniques that could be directly applied on numeric sequences, including Wavelet transform [109, 110], Fourier transform [111], and time series shapelets [112].

We proposed a compressed representation of instances based on the quartiles of the duration distributions in [113]. This data representation is used as a basis for the discovery of duration-related patterns in symbolic time series. It reduces the input size of the dataset before going to the learning procedure, but at the cost of losing some information in the original dataset.

An alternative way is to design distributed versions of existing algorithms or framework. Unlike dimensionality reduction techniques, distributed solutions keep the integrity of data instances and take advantage of available computation resources to accelerate the computation process [114]. In principle, distributed infrastructures make it possible to import all data in memory, parallelize existing sequential algorithms, and independently process individual components. One distributed system named Storm [36], that is distributed and fault-tolerant real-time computation, is widely used for the purpose currently. We propose a systematic way of distributing collective dynamical modeling clustering algorithm using Storm to extend the capability of processing large-scale dataset of time series in chapter 5.

In this chapter, we introduce Apache Storm (a distributed and fault-tolerant real-time computation platform), its basic concepts (spout, bolt, streaming groupings), operating mechanism (topology and configuration) in section 5.1, discuss the deployment plan of the CDMC framework using Storm in section 5.2, and prove the efficiency of distributed CDMC framework in section 5.3.

5.1 Apache Storm

Storm is an open source of distributed real-time computation system. Storm can be used to analyze streams coming from a data source [36]. Compared with Hadoop [115], which uses for batch processing, Storm can process streaming data in real-time. In Storm, there are five primitives:

- *Streams.*

  Stream is data abstraction in Storm. It is defined as an unbounded sequence of tuples that could be processed in parallel. Every stream is identified by a id.

- *Spout.*

  A spout is a source of stream that reads data source and emits a tuple (a named list of values) as input to the next bolt. Spouts can emit more than one stream.

- *Bolt.*

  A bolt is a processor in the topology, receiving a tuple from a spout, handling and emitting it to the next bolt. Bolts can emit more than one stream.

- *Groupings.*

  A grouping defines the partition mode of tuples in bolts. There are eight groupings, including shuffle grouping, fields grouping, partial key grouping, all grouping, none grouping, direct grouping, local or shuffle grouping. As illustrated in Figure 5.1, direct, shuffle, and all groupings are specified in our distributed plan.

- *Topology.*

  A topology consists of spouts and bolts. Each one is similar to a Map-Reduce job but runs forever, unlike that Map-Reduce job eventually finishes.

The many-to-many relationship between spouts and bolts leads to its flexibility of distribution across clusters in a network. It is easy to define a customized topology in terms of criteria users predefine (tuple groupings) in bolts. Among advantages are that each bolt in a topology will distribute the workload based on the desired parallelism parameter set up in Storm's configuration. It also forces a predetermined number of threads working together for the same task. If a bolt fails to finish a job, Storm will automatically reassign the failed job to other bolts or machines. Therefore, Storm guarantees that there will be no data loss, even if machines go down and messages are dropped out. Figure 5.1 is an example of a topology consisting of a spout and three bolts.

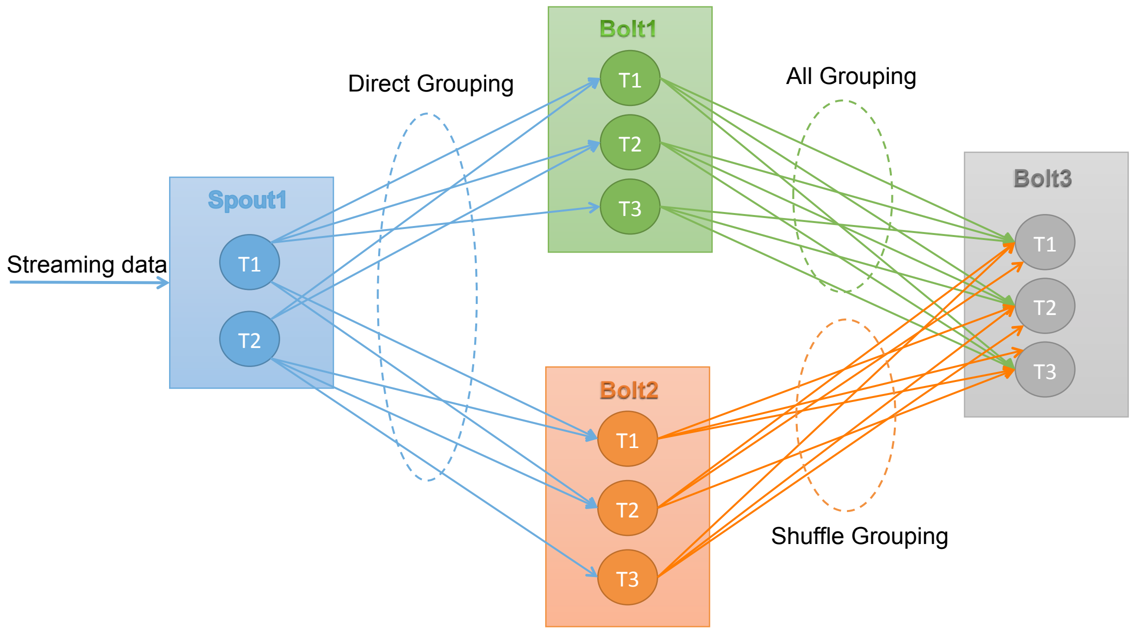


Figure 5.1: Example of a topology of one spout and three bolts. Each spout or each bolt has multiple tasks assigned by default or users. A spout connects to two bolts and a bolt receives tuples from different streamings. Different streaming groupings indicate different ways of sending tuples between sets of tasks. In Storm, there are eight streaming groupings and here only three of them are used. Direct grouping means a spout or a bolt decides which bolt receives a tuple emitted by the spout or the bolt. All grouping lets a stream be replicated across all bolts in a topology. Shuffle grouping refers to randomly distribute tuples across the bolt's tasks with load balance.

5.2 Distributed Deployment on Storm

Collective Dynamical Modeling & Clustering (CDMC) is a sequential algorithm. However, it is not good at dealing with large-scale dataset. The complication would worsen if Expectation Maximization (EM) algorithm were applied in training dynamical models due to a big computation complexity brought by EM itself. The running time of CDMC is naturally high as there is a big number of times series of long length. To solve this scalability problem, we focuses on partitioning the input sequences on a certain way so that individual partitions run in parallel and independently. To accomplish this, it is necessary to analyze internal structures of the CDMC framework and distribute them in Storm with powers of flexibility of processing streaming data as illustrated in Figure 5.1.

5.2.1 Distributed CDMC System Design

The systematic deployment of the CDMC framework on Storm is briefly depicted in Figure 5.2. There are four steps of allocating individual components of the original version of CDMC on Storm. The general idea is to avoid sequential arrangement of the input time series and decompose CDMC into a parallel architecture. To specific, the number of bolts for storing cluster results is the number of distinct cluster labels, and the number of bolts for modeling data likewise. The main design details are listed as follows:

- For the data source, it is necessary to cache a desired number (e.g., n) of time series before starting procedures of modeling and clustering input data. When n time series are ready, data source spout gives out each time series into an assigned bolt in terms of its cluster label. Initially, cluster labels for an input time series are randomly specified. After that, the clustering results produced by clustering step in Algorithm 1 are applied to the existing time series.

- For the distribution of the input time series, time series with the same cluster label are stored in the same bolt. For each bolt, it leaves room for manipulating local storage of time series before they are sent to the next modeling bolt. For example, discretizing time series and compressing time series in a further step.

- For the modeling step, each bolt receives tuples (time series) from the corresponding bolt in the clustering step, then builds up a model on the data collection. After that, it produces model parameters or takes models as input to the next clustering bolt.

- For the clustering step, based on the trained dynamical models, group each time series into some cluster in terms of the predefined criterion, such as maximum likelihoods of the input data using currently trained models.

Note that if a new time series comes in, it would be temporarily put in the buffer of data source bolt in Figure 5.2. Until the next clustering results arrive in the bolt, the new time series could be added into the dataset with randomly selected cluster label. To extend the parallelism of the CDMC framework, it is able to parallelize the local modeling procedure, provided that the modeling process could be divided into several independent parts.

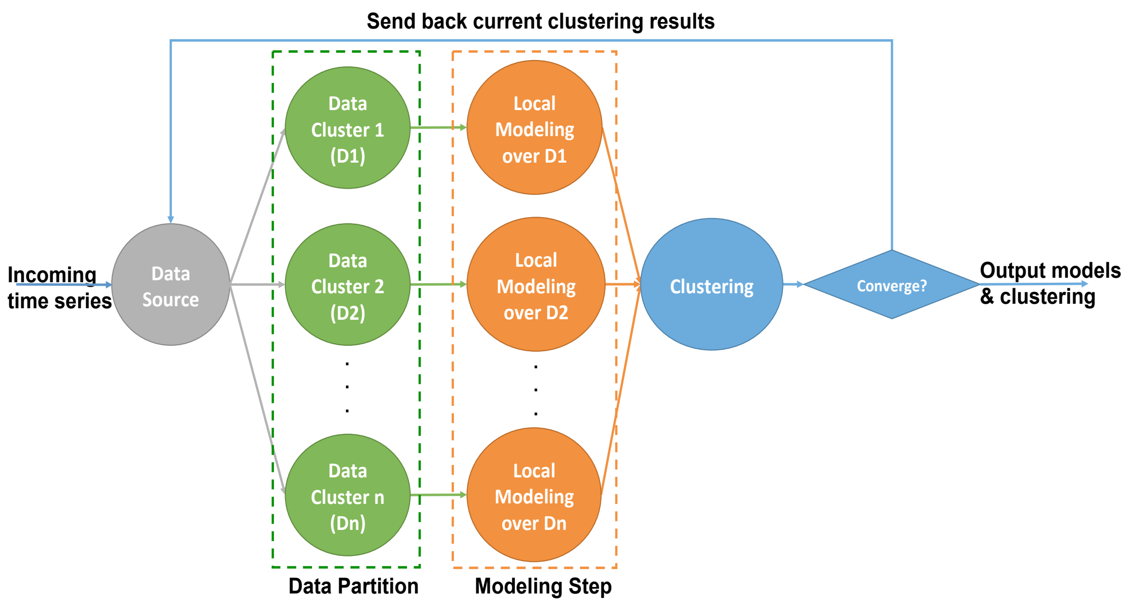


Figure 5.2: Distributed Collective Dynamical Modeling & Clustering Framework Design. Distributed CDMC framework design determines if two consecutive clustering results are similar enough to each other after each clustering step is finished. Thus, it is the occasion when the given data are grouped into data cluster bolts in parallel. Each data cluster bolt works as input to the corresponding local modeling bolt. The results in local modeling bolts go to the clustering bolt that emits the current cluster labels back to data source bolt if converge not. The cycle never stops until convergence.

5.2.2 Distributed CDMC System Implementation

There are six components in distributed collective dynamical modeling & clustering system implementation as listed in Table 5.1:

Table 5.1: Distributed Collective Dynamical Modeling Clustering System.

Distributed Collective Dynamical Modeling Clustering Components	
Core Components	Initial Data Assignment Module Data Partition Module Data Modeling Module Evaluation Module
External Components	Data Source Module System Parameter Configuration Module

In Table 5.1, "Initial Data Assignment Module" corresponds to clustering time series in chapter 2; "Data Partition Module" corresponds to clustering step in CDMC framework; "Data Modeling Module" corresponds to dynamical modeling in chapter 3, and "Evaluation Module" corresponds to stopping criteria in chapter 4. For "Data Source Module" and "System Parameter Configuration Module", they work as external parts in collective dynamical clustering modeling framework, respectively for giving out data instances and making initial configurations in this distributed system. The details of each component in the system implementation are presented as follows:

Initial Data Assignment Module: This module is responsible for providing initial cluster labels of data instances as input to the distributed system. As introduced in chapter 2, not only do those cluster labels predefine data partition in cluster assignment, but also reduce the probability of locally maximizing parameters in modeling steps. In this system implementation, this module acts as an independent part of distributed collective dynamical modeling clustering system. The rationale behind this design are two folds:

- Dynamic time warping computation.

As introduced in section 2.1, dynamic time warping is a dynamic programming technique that provides an optimal alignment between two time series by nonlinearly warping their time dimensions. Due to the nature of dynamic programming optimization, the current optimal warping path should be derived from paths ending in the previously consecutive positions. Dynamic warping path between any two time series are different from that of another two time series. There are no sharing paths in those two pairs of time series. Even worse if a majority of time series are distributed uniquely among them. Thus, there should be no need to make an distributed version of dynamic time warping computation. Neither variants of dynamic time warping in section 2.1 do, including deviation-based dynamic time warping [35].

- System resources limitation.

  In reality, there always are a limited amount of computation resources. On the one hand, these available computation units mainly focus on processing data (clustering and modeling sequences in our work). Except those allocated resources, this is the case that few amount of computation units are left to do initial data assignment task. On the other hand, most of time it is impossible to do dynamic warping over all input data because of the uncertainty of incoming data. An initial data process is executed on only a small amount of data instances. Had the distributed collective dynamical clustering and modeling framework started, this module would exit forever. In a word, available resources are mainly used to speed up computation in our distributed framework, rather than being allocated in dynamic time warping calculation.

Although it is not recommended to integrate initial data processing module into the distributed system, there is an approach to accelerate initial data processing in virtue of big data distribution. The principle is that data instances are loaded into bolts (basic unit in Storm for processing data) according to uniformly randomized deployment. Then each bolt conducts dynamic time warping computation over all possible pairs of instances in it. The computation results are sent to a global bolt (independent from all bolts running dynamic time warping calculations), which store results of dynamic time warping from all other bolts. Had a bolt finished its tasks, it would communicate with other bolts to get access to their instances to continue calculations of dynamic time warping between its instance and instances from other bolts. Until all pairs of all instances in bolts are processed, results in the global bolt will be sent to a new bolt that executes clustering algorithm described in section 2.2 so as to produce initial cluster labels for the input data instances.

System Parameter Configuration Module: This module deals with system parameter estimation, including bolt allocation (clustering, modeling, and evaluation tasks), cache size (memory and disk spaces) in each bolt, and parallelism (number of processors) in each bolt.

- Bolt allocation.

  A topology consists of spout and bolt units. In our system, there is just one spout emitting tuples coming from external data source. However, the total number of bolts is determined by clustering, modeling, and evaluation tasks theoretically. For clustering tasks, the number of bolts depend on the number of unique cluster labels and percentages of instances over all instances in clusters. For modeling tasks, specific models specify the number of needed bolts. For instance, each semi-Markov model in section 3.2 has two parts: state transition and state duration distribution. Thus, there are at least two bolts corresponding to each of two parts in each semi-Markov model. For evaluation, one bolt is big enough to hold current clustering results and execute comparison with the previous one. Note that the actual number of available computation units decides bolt allocation in distributed system. This is possibly the case that two or more bolts might be deployed in the same computation resource.

- Cache Size.

  It is meaningless to calculate cache size for computation unit, if there are unlimited memory and disk spaces to save input data instances and intermediate results. Given the fact of finite computing resources (memory, disk space, and CPUs), there thus is a necessity of estimating cache size up to capacities of computation pieces. In addition to the limited computation power, the size of input data instances, the total number of bolts, and intermediate results during modeling steps are previously derived. In terms of these factors, the cache size in memory for one computation unit, for example, is estimated in equation 5.1.

$$
\begin{aligned}
cache\ in\ memory = input\ data\ instances * \frac{number\ of\ bolts\ in\ a\ computation\ unit}{total\ number\ of\ bolts} \\
+ intermediate\ results
\end{aligned} \tag{5.1}
$$

- Parallelism.

  For each bolt, it is guaranteed that there is at least one thread of operating it in Storm. If there are still available threads, they are successively assigned into bolts of modeling data instances with the highest priority. Then the remaining threads are allocated in clustering step. For data source and evaluation modules, one thread satisfies sending out instances and comparing clustering results.

Note that since bolts in Storm are able to process streaming data from different spouts, it is possible in our system to apply for more than one spout to emit data. We assume that it is big enough for one spout to hold all of instances and the number of spouts is set to be one by default. If our system detects spouts fails to send out tuples, we could apply two or more spouts in a new start-up computation.

Data Source Module: This module aims at sending out data instances given a data source stored in disks. In Storm, a spout is a source of streams in a topology. When the system starts, a spout will read user-defined tuples from the external source such as data instances in disk and emit them into the topology. Besides data resided in disk, there are following data stored in this module:

- Used Instances.

  Used instance refer to the existing instances which has been grouped into clusters in the system. Generally, there are two types of them: (i) initial instances processed by "Initial Data Assignment Module" to start up the distributed system and (ii) instances that come after those initial instances but have been processed by bolts.

- Newly Incoming Instances.

  Newly incoming instances are those loaded into system but being processed by bolts after initial instances have been used to do clustering and modeling tasks in the system.

- Global Grouping Table.

  Global grouping table records the clustering info of data instances and bolt allocation for data instances, which would be sent out to all bolts directly connected to the spout. Due to the popularity of global grouping table in the entire system and big amount (e.g., millions) of instances, a bit array in its implementation is taken to reduce its space so as to achieve fast transmission between bolts.

The procedure of data source module is: when a newly incoming instance comes, it is first stored in cache. Until a predefined number of new instances are arrived, they would be send out to the neighboring bolts. This strategy not only allows to give a certain amount of time to data modeling module to build up models before updating them, but also takes care of the case that few number of recent data takes no effect on forming new clusters from the past clustering results. When the global grouping table is received, the spout checks the new table with the current one in it, finds the difference between them, and emit tuples only in the new table. The difference indicates the number of used instances to be resent because of the absence of instances from destination bolts.

Note that at the beginning of running a topology, its spout spends lots of time on sending out used instances due to a big update from the received table. After several rounds of clustering and modeling modules, the difference is so small that it is fast to process of sending used instances.

Data Partition Module: This module focuses on partitioning data from data source module and extracting features from data instances, and then deliver features to connected bolts.

- In partitioning data, it is assumed that bolts in this module are dependent from each other and there are communications between them. In other words, when an instance is needed but not in the current node, it is able to get access to the needed instance from other partners. The reason behind this design is that given bolts allocation in the received global grouping table from data source module, a bolt information that an instance belongs to which of bolts is known. The current node with missing instance sends a request to the designated node, which gives out the needed instance in return. Also, the utilization of CPUs is taken into account. It is ideally to put CPUs in execution most of time and hold a performance balance between clustering and modeling modules.

- The features depend on model types. For example, in semi-Markov chain model, extract state transition and state duration as features and send them out to modeling bolts. Each instance is represented by a vector of features, which is outputs of clustering bolts.

The procedure of data partition module is: when receiving the global grouping table from data source module, a clustering bolt compares the received table with the existing table to find out missing instances. The bolt requests all missing data from other clustering bolts. When a missing instance is arrived, extract its features and cache them. When all missing instances are processed, the bolt outputs feature presentations to neighboring modeling bolts. Note that when the distributed system is initiated, all instances are missing in clustering bolts because of no instance cached in it. After the first round of clustering and modeling parts, instances cached in clustering bolts might be hit and thus there is no need for clustering bolts to extract features from those instances. Therefore, cache results reduces time of processing delivering instances.

Data Modeling Module: This module builds up models given features from clustering module and evaluates them based on trained models. This module guarantees integrity of training data both from clustering modules and cached data. While training model, bolts independently work by themselves. In evaluation step, bolts communicate with each other to evaluate their own models on all instances bolts have. Evaluation results depend on types of models. For example, semi-Markov chain model saves negative log-likelihoods of data instances.

The procedure of this module is: when a global table arrives from data partition module, modeling bolts start checking needed data in their caches, while receiving data from data partition module. when data is ready, a modeling routine is initiated. Initial model parameter is randomly chosen if there is no definition in advance. Otherwise, previous modeling results do. When one modeling bolt itself finishes evaluating instances, it sends requests to other modeling bolts get all other instances that are not present in it. All other bolts receiving the requests send back requested data in return. After the modeling bolt is done evaluating all instances, evaluation results are transmitted to evaluation module. Finally, when all of modeling bots output results, the evaluation module begins grouping data task.

Evaluation Module: This module is in charge of two tasks: (i) group data instances into a designated number of clusters and (ii) stop running the topology if comparison of current clustering results with the previous one is close enough to each other. There is one bolt working on the two tasks in our system implementation. The procedure is that when evaluation results from data modeling module are arrived, each instance is assigned to one of clusters with the best one. All instances get clustered and then the bolt compares the current clustering results with the previous one. If they are close to each other, it stops running the system and produces the final clustering and modeling results. Otherwise, the evaluation bolt updates clustering results and sends back a copy to data source module. Note that evaluation results may be cached since it provides another way of defining stopping criteria, setting up a threshold value for all negative log-likelihoods of instances for example.

The mechanism of the above six components works as follows: “Initial Data Assignment Module” labels data instances as input to “Data Source Module”. “System Parameter Configuration Module” estimates initial configurations in the distributed system according to data instances, constructs a topology from the estimation, and configures parameters before our system launches. “Data Source Module” loads data with cluster labels stored in a global grouping table and emits them into connected clustering bolts. Then “Data Partition Module” partitions data based on cluster labels in the global grouping table from “Initial Data Assignment Module” in the first round or a new global grouping table in the following rounds. After getting data partitioned, “Data Modeling Module” builds up dynamical models and evaluates input data instances. Next, “Evaluation Module” groups input data and compares the current grouping results with the previous one. If they reach a stable status, it produces the current grouping results and dynamical models as final results. Otherwise, it updates the global grouping table and sends a copy back to “Data Source Module” to start up a new round of computations. The role of a global grouping table is considered as a signal to initiate each components in the distributed system.

Apache Storm supports dynamic distribution of computation resources to some degree. However, it requires that when one of bolts fails, it can start itself over again, apply for a new bolt to replace the bad bolt, or even stop the current run and re-distribute computation sources. All solutions have a very long response time. Thus, it is necessary to respond such bad case in a short time. The past experiences in experiments proved that the effect of adjusting parameters in Storm is not a good choice, relative to the requirement of adapting fast re-clustering and re-modeling processes. Even worse if that Storm promises fault tolerant and fail fast features. Both of solutions need extra computation sources to be auxiliary sources to be prepared for data recovery. Therefore, a certain number of resources are wasted while waiting for Storm to make use of them. So this way of dynamically allocating resources could be considered as a passive one on the system level.

On the software level, it is flexible to detect general usage of computation units and thus respond it with dynamic distribution of sources. In principle, we can detect heavy workload indicated by the number of instances in each cluster bolt (bolts in a cluster are guaranteed to be workload balance). To collect such global information of instance distribution, there needs to gain a general distribution table with exact number of instances in each bolt. In each bolt, the system locally collects the number of instances in it and reports it to the distribution table within a time interval. If we detect a bad case of unequal distributions of instances, we could move new data into a bolt with light workload if possible. This dynamic scheduling method could prevent from failing running tasks and thus solve the proposed problem in a fast responding time. Although at the cost of additional requirement of extra resources, it is meaningful for the system to add such detecting component.

Overall, the systematic deployment of distribute collective dynamical modeling clustering framework on Storm is illustrated in Figure 5.4.

Table 5.2: A summary of caches in distributed CDMC system

Module Name	**Cache Functionality**
Data Source	1. Used instances including initial instances and others in bolts. 2. Newly incoming instances not processed by bolts. 3. A global grouping table.
Data Partition	1. Feature representation of data instances. • A state transition of data instance. • A state duration of data instance. 2. A global grouping table.

Continued on next page

Table 5.2 – *Continued from previous page*

Module Name	**Cache Functionality**
Data Modeling	1. A set of modeling parameters. 2. Features information. 3. A global grouping table.
Evaluation	1. Evaluation results. 2. A global grouping table.
Note	Caches in collective dynamical modeling & clustering framework refer to available memories and external disks in computation unit such as computers, nodes in clusters, and so on.

Compared with sequential system, the main factors that determine efficiency gains due to parallelization are as follows:

1. Intermediate results caching. The intermediate results as shown in Table 5.2 include clustering results for instances, extracted features of instances (for semi-Markov chain model, state transition matrices and state duration matrices collected from dedicated bolts in modeling step), and the parameters derived from the models. Except the current clustering results saved in the sequential system, others are only stored in the distributed system. These cached information reduces the processing time in the next rounds, rather than being recalculated in the sequential system.

2. Multiple task processing. For distributed system, it has advantages of processing multiple tasks in parallel and allocated resources (memory, hard disk, registers in processors) reasonable for dealing with multiple tasks in parallel. Even if sequential system shares the same resources as the distributed system does, the internal parallelized architecture only maximizes the usage of processors, reduces disk I/O, and memory consumptions. In Storm, it is capable of allocating many bolts to do one task (clustering or modeling processing components) and the resulting computation power is greatly enhanced. Therefore, such mechanism speeds up outputs of the entire collective dynamical modeling & clustering framework.

5.3 Experimental Evaluation

We evaluate distributed collective dynamical modeling clustering framework through time limit of processing modeling and clustering instances, as a metric to compare performances of sequential system of collective dynamical modeling & clustering framework and the distribution solution.

5.3.1 Experimental Setup & Methodology

Two real datasets are used in experiments: human sleep patterns [59] and web user navigation behaviors [31]. Experiments on web user behavior dataset are much different from those on human sleep. The size of human sleep dataset is much smaller than web user behavior. When loading web user behavior datasets into distributed system, it is reasonable to put a predefined number of instances in the system and then when the system could accommodate more instances, new instances from the dataset are entered into the system bunch by bunch. Such deployment of instances could be easily to record and observe the performance differences between sequential and distributed systems with the increasing number of instances.

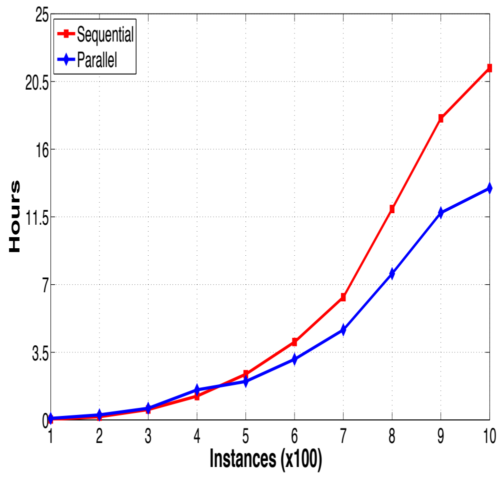
In web user behavior dataset, there are over a million instances in the dataset. It is possible to duplicate instances in the dataset to test a maximum processing capacity for distributed system. To make the comparison of experiments reasonable between sequential and the distributed system, it begins with a small amount of instances in the system. The small dataset could evaluate the time interval of modeling and clustering in the sequential system and then more instances are loaded into the system. It is possible that the sequential system could not process a large amount of instances since it exceeds the capacity of computing devices, unlike the distributed one that distribute such amount of instances into various computing devices to extend the computing capacity.

Another difference between these two datasets are parameter setup (the number of states in dynamical model, the number of bolts in clustering and modeling modules, for example) in web user behavior dataset. There are 17 states (the number of categories of web pages in the entire dataset). There are at least five bolts of modeling instances in the distributed system and there are another set of 5 bolts for clustering instances. The topology of all nodes (including bolts and spouts) is more complex than the structure of five bolts in human sleep dataset. Due to a large deployment of computing nodes in Storm system, it takes a long time (in hours or in days) to processing the final results of dynamical models (hidden Markov models and semi-Markov models) and clustering results.

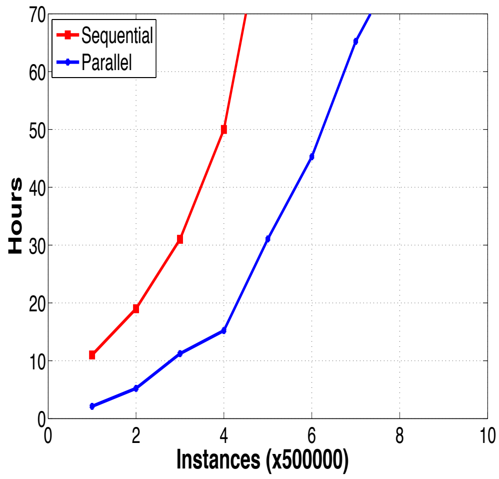
For the sequential system, there is a single machine with one processor and a 1-Gigabyte memory and 100-Gigabyte disk space to store web user behavior dataset as well as models and clustering results in the device. For the distributed system, there are three machines with one processor for each machine and the same size of 1-Gigabyte memory and 100-Gigabyte disk space to hold data instances as well as models and clusters. The dynamical models are semi-Markov chains.

5.3.2 Time Performance

In Figure 5.3, for human sleep patterns, time performance has not a big variance between sequential procedure and parallel procedure at the beginning of loading a small amount of instances. For web use navigation behavior data, the number of bolts for modeling instances is set to be 5 compared with each clusters with 17 states. The number of log instances increases from 500000 to 4000000. The sequential system has a very long time of modeling and clustering at most 2500000 instances. It is a big workload for the sequential system and its entire computation time is beyond the time spent in the parallel system that processed 3500000 instances under 70 hours. Thus, a substantial reduction in processing time is observed in the distributed (parallel) version as compared with the sequential algorithm as shown in Figure 5.3. The performance improvement increases noticeably as the number of instances grows in web user navigation dataset.



(a) Human Sleep Pattern



(b) Web User Navigation

Figure 5.3: Time performances of human sleep patterns (the left subplot) and web user behavior navigation (the right subplot) datasets in distributed collective dynamical modeling & clustering system. With the increase of data instances, the distributed system outperforms the sequential one in processing input data. Note that due to the relatively small number of human patterns, at the beginning of experiments there is a twist of performances in the two systems. When more data are loaded into the distributed system, it gives a substantially decrease in processing time.

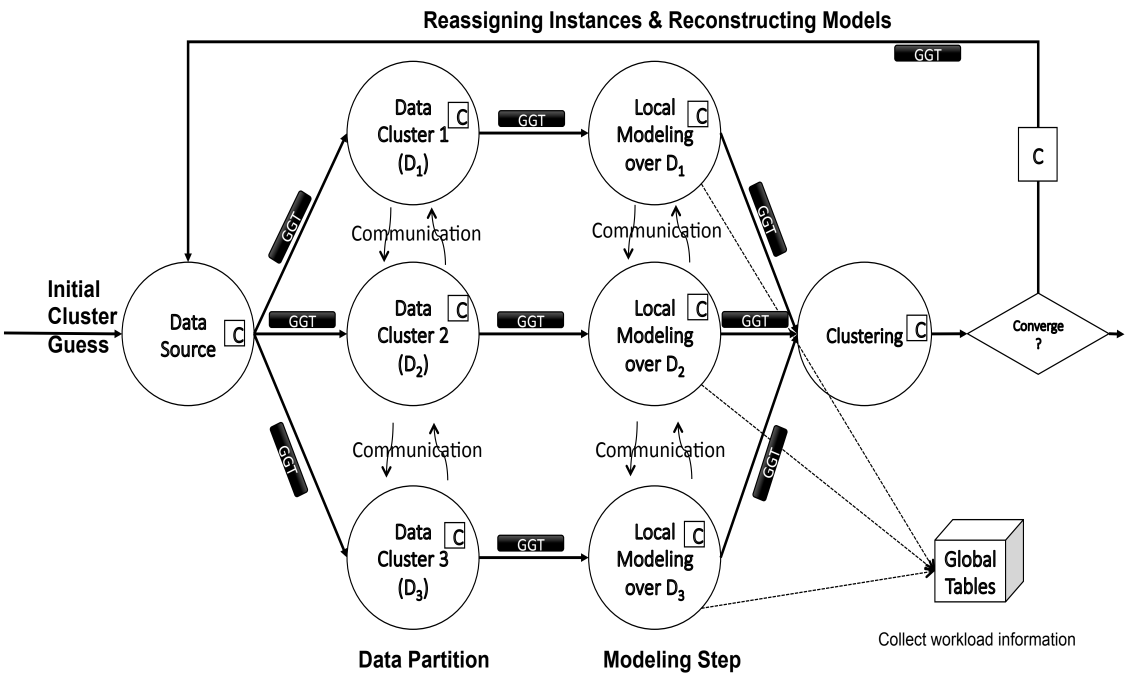


Figure 5.4: Distributed Collective Dynamical Modeling & Clustering Framework Implementation. Take three models and three clusters as example. When incoming sequences come into the system, they first are loaded into data source bolt, and then grouped into the corresponding clustering bolt according to initial cluster label guesses. Then for each group, there would be a dedicated bolt to build one model over data. After finishing modeling process, all of instances in all three bolts would be reassigned to clusters according to the currently created models. Finally, if two consecutive clustering results are close enough to each other, the system would stop; otherwise the clustering-modeling process would continue until a stable clustering result is produced or a maximum number of iteration is reached. A global grouping table (GGT) is considered as a signal to initiate each components in the distributed system.

Chapter 6

Applications

In this chapter, we apply our semi-Markov dynamical model version of CDMC to the human sleep dataset described in section 6.1 and to the web user navigation dataset described in section 6.2. We present experimental results and analysis.

6.1 Human Sleep Patterns

Human sleep can be described by individual sleep stages determined by physiological signals obtained through full-night polysomnography. The signals used for sleep staging include brain electrical signals (electroencephalography, EEG) and eye movement signals (electrooculography, EOG). Stages correspond very roughly to different depths of sleep. There is an alternation among stages during the night shown in Figure 6.1, though not a deterministic one, and not one that is yet well understood by researchers.

Human sleep patterns are closely associated with overall health and quality of life, making the scientific study of sleep an important pursuit. Sleep stage composition is a basic description of sleep structure that comprises total sleep time, sleep efficiency, and percentage of sleep period time in each of the stages within a night of sleep [116].

However, these features provide an incomplete description of human sleep that does not capture the dynamical information in hypnograms.

Sleep stage transitions [11] and bout durations [10] shown in Figure 1.1 are essential indicators in characterizing the structure of sleep. Typical patterns of human sleep have been found [9], yet sleep microstructure varies across individuals, being affected by age, circadian rhythms [117], and other factors.

A substantial challenge in modeling the dynamics of sleep is the scarcity of key dynamical events including stage transitions within sleep sequences as shown in Figure 1.1. This scarcity yields small samples over which dynamical models are to be trained, leading to high uncertainty in parameter estimates. An approach known as dynamical modeling-clustering (CDMC) was proposed [34] to address this challenge. CDMC reduces model variance through selective aggregation of instances during a clustering phase, so that models are learned over collections of dynamically similar instances rather than individual instances. The following section shows that CDMC initialization using clustering by Dynamic Time Warping (DTW) similarity [51, 57] yields good convergence properties [59].

We focus on modeling the statistical and dynamical characteristics of sleep, both the transitions between stages and stage durations. Using full-night sleep stage data for several hundred patients, it is going to extract statistical information about sleep stage durations and transitions, with a goal of developing new statistical machine learning approaches to modeling the dynamics of human sleep.

The main objectives in human sleep patterns are displayed in the following three directions:

- Study empirical statistical properties of human sleep stage dynamics, particular stage transitions and stage durations.

- Propose suitable non-stationary statistical (dynamic) models for human sleep, which aims at clustering.

- Investigate the potential of using the proposed model in the discovery of relationships between sleep dynamics and overall health.

6.1.1 Data Description

The human sleep dataset consists of a total of 244 fully anonymized human polysomnographic recordings. They were extracted from polysomnographic overnight sleep studies performed in the Sleep Clinic at Day Kimball Hospital in Putnam, Connecticut, USA. This population consists of 122 males and 122 females, all suffering from sleep problems. The subjects' ages range from 20 to 85 and the mean value is 47.9. The polysomnographic recording is extracted from C3-A2 EEG time signals at a sampling rate of 200Hz and then staged by lab technicians at the Sleep Clinic [116]. Staging of each 30-second epoch into one of the sleep stages (wake, stage 1, stage 2, stage 3, and REM (Rapid Eye Movement)) is done by analyzing EEG, EOG and EMG recordings during the epoch [79]. In this section, stages 1, 2, and 3 may be grouped into a non-REM stage (NREM) and stages 1, 2, and REM be combined into light sleep stage. We directly use human hypnogram recordings to capture dynamical features of sleep. The durations of continuous uninterrupted bouts in each stage are natural candidates for feature representation of the hypnogram recordings.

In sum, three versions of the human sleep datasets are considered, depending on whether these stage labels are grouped in some way:

- (W5) uses the five standard stage labels Wake, 1, 2, 3, REM.
- (WNR) uses the three stage labels Wake, NREM (stages 1, 2, and 3), REM.
- (WDL) uses the three stage labels Wake, Deep (stage 3), Light (stages 1,2,REM).

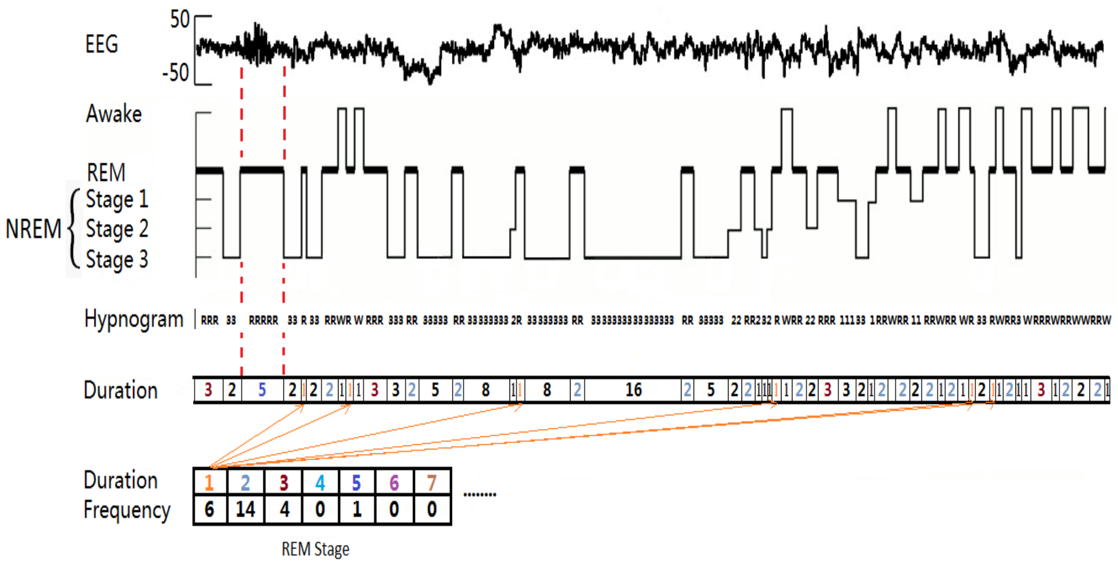


Figure 6.1: EEG is used to record the electrical activities of participants in sleep research during night, which measures voltage fluctuations of current flows as a cap with multiple electrodes installed on the human scalp. The raw electric signal is then discretized by sleep experts in a finite number of stages, which could be categorized into wake, stage 1, stage 2, stage 3, and REM stages. A sequence of stages in a person's sleep during a full night is called hypnogram. A continuous subsequence of a stage in a hypnogram is called duration of the stage and labeled by its length, the number of stages that appears in the subsequence. Furthermore, the frequency of duration of one stage in the hypnogram is collected in a table is represented as statistical property of human sleep.

6.1.2 Dynamical Model Evaluation

A comparison between sequences generated by each the Markov chain models (MM) and semi-Markov chain models (SMM), and between these and the original dataset hypnograms, shed additional light on the stage transition dynamics captured by each of the models. Figure 6.2 shows an original dataset hypnogram, simulated stage sequences generated by the trained models, and a randomly generated sequence. The randomly generated sequence was obtained by assigning a randomly chosen sleep stage to each epoch using a uniform distribution over the sleep stages. The typical hypnogram shown, s, was selected from the dataset, such that its generative log likelihoods in the Markov chain model and the semi-Markov chain model, $P(s|MM)$ and $P(s|SMM)$, are close to the corresponding average log likelihood values over the entire dataset, which are approximately -148 and -152, respectively.

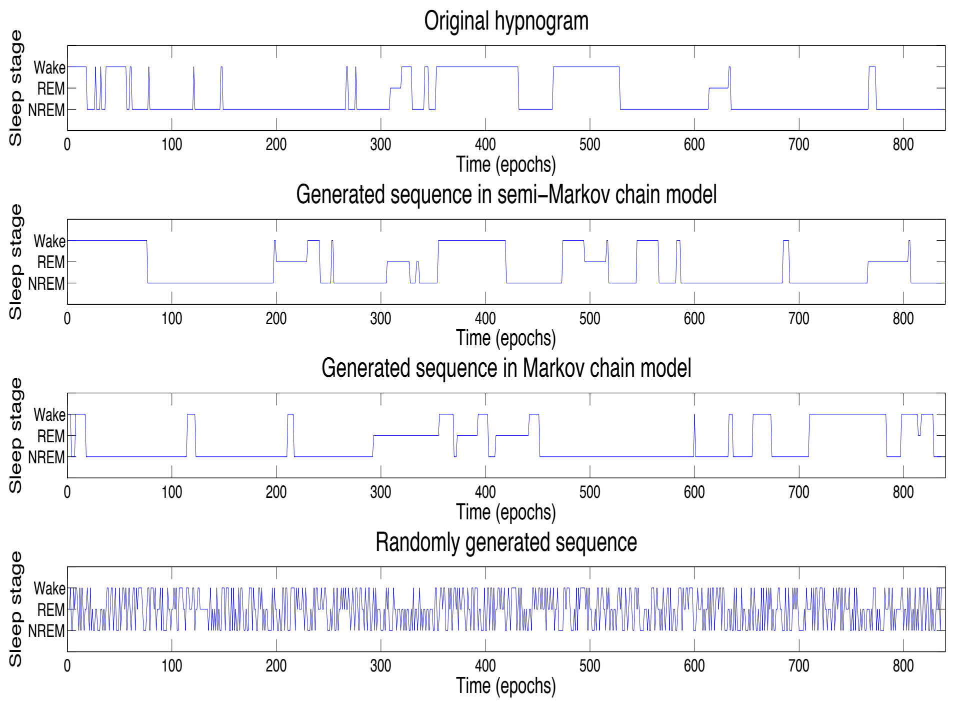


Figure 6.2: Comparison of an original dataset hypnogram, hypnograms generated by Markov chain and by semi-Markov chain models, and a randomly generated hypnogram.

Figure 6.2 shows, especially near the middle of the night, that the SMM better captures the frequency of both short and long wake durations observed in the original hypnogram, as compared with the MM. At the onset of the night, the long uninterrupted duration of wake stage in SMM simulates the original hypnogram for the same time period better than the short durations of wake stage in the MM sequence do. Although MM is comparatively worse than SMM, its overall distribution of stages matches the all-night stage composition of the original hypnogram more accurately than the random sequence does. This single example is of course intended only as an illustration. The more general analysis of section 3.3 makes clear the superiority of the SMM over the MM as a model of stage bout durations, in a robust statistical sense.

6.1.3 CDMC Framework Evaluation

The collective dynamical modeling clustering algorithm was used for clustering, with semi-Markov chain models as the dynamical models. Initial cluster guesses were computed by deviation-based DTW clustering as described in Algorithm 4, with either gwDTW (Equation 2.5) or sdDTW (Equation 2.6) as the distance metric. Clustering driven by the standard DTW (Equation 2.4) distance metric was used as a basis for comparison.

Clusters Quality: Due to the long average duration of a given stage in human sleep, it is better to explicitly describe its distribution of the sleep stage. In contrast with the exponential one (as shown in Figure 3.6) in hidden Markov models (HMM), Figure 6.3 illustrates the CDMC clustering results for the semi-Markov dynamical models with Weibull state durations over 100 randomly chosen instances in the human sleep data. The coordinates of each instance are the estimated parameter values of the Weibull distribution for that instance's wake duration distribution. Cluster centroids are significantly separated along both parameter axes ($p < 0.05$). The resultant two-sample t-test over coordinates of clustered instances in Table 6.1 of instances proves the significantly difference of separating those two clusters in Figure 6.3.

Table 6.1: T-test of coordinates on human sleep dataset.

	H value	**P value**
Param1	1	0.0017
Param2	1	0.0092

Models Analysis: Generative negative log likelihood was used to measure the quality of model fit for unsupervised clustering. Given a dynamical model, M, built over a

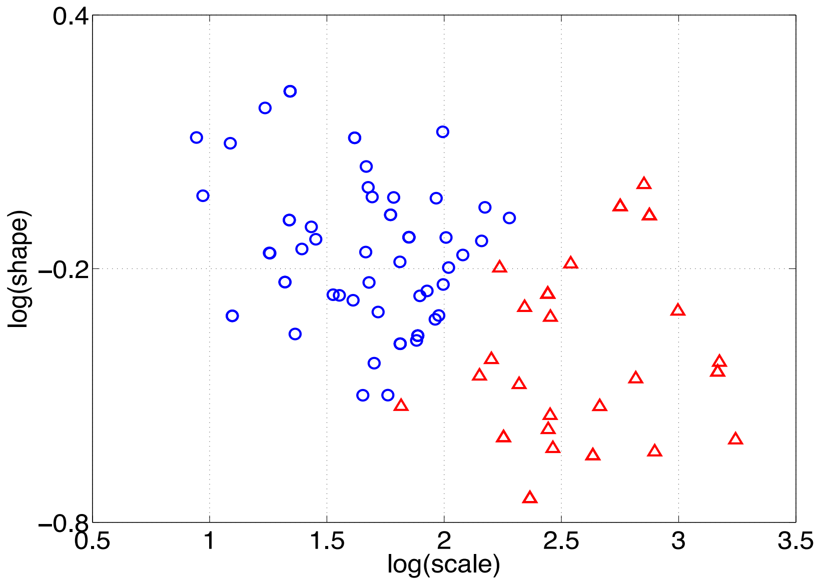
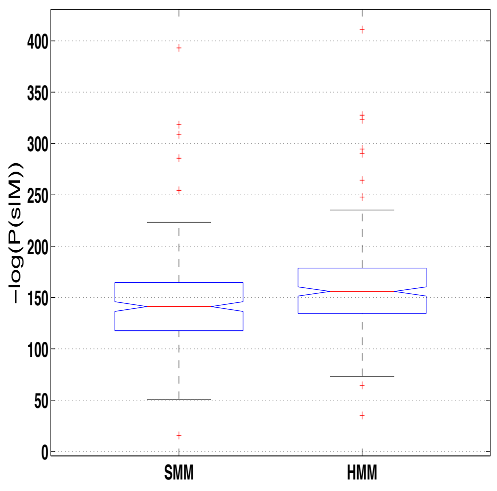


Figure 6.3: Visualization of CDMC clustering of human sleep data with Weibull semi-Markov dynamical model. Coordinates of each instance are parameter values of wake stage Weibull model fit individually to the instance.

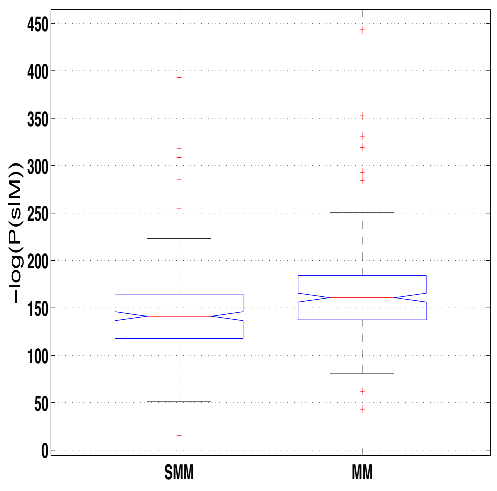
group of sequences such as human sleep sequences, the generative negative log-likelihood $-log(P(s|M))$ of a sequence, s, is a measure of the probability that the sequence, s, would be produced by the model, M. Lower negative log-likelihood values (higher generative probabilities) imply a better model fit. The goal of clustering was to minimize the generative negative log likelihood. Comparison of median negative log likelihoods for different models was measured by a Wilcoxon rank sum test as described in section 2.4.

To investigate the importance of models in analyzing data, the human sleep data (WNR version) were clustered using CDMC with two clusters, for each of three dynamical model types: semi-Markov chains with Weibull state durations , hidden Markov models, and Markov chains. Three-state chains were used in all cases. The generative negative log-likelihood $-log(P(s|M))$ was used to measure the quality of model fit. Figure 6.4 shows the results. The median negative log-likelihood of the semi-Markov version is significantly better than that of the Markov chain version ($p < 0.05$, Wilcoxon-

Mann-Whitney test). Comparison of the semi-Markov version of CDMC against a hidden Markov model version also resulted in superior performance of the semi-Markov version ($p < 0.05$).



(a) SMM VS. HMM



(b) SMM VS. MM

Figure 6.4: Negative generative log-likelihoods of CDMC clusters for semi-Markov (left) and hidden Markov (right) dynamical models in Figure 6.4(a), as well as semi-Markov (left) and standard Markov dynamical models (right) in Figure 6.4(b). Non-overlapping notches indicate significant difference in medians ($p < 0.05$). Semi-Markov models provide significantly better log-likelihood than other two models.

To fully examine dynamical models and various versions of hypnograms described in section 6.1.1, three Markov models were respectively tested in pairs as dynamical models in the CDMC framework. The t-tests of generative log-likelihoods (logarithmic values of probabilities of generating hypnograms given each Markov model) between each pair of models are shown in Table 6.2. Semi-Markov chain models with Weibull state durations perform significantly better than other models in terms of generative log-likelihoods on instances of hypnograms. Markov chain model and hidden Markov model imply exponential distribution of stage bout durations while semi-Markov not. With the increase of the number of stages from three to five in data, semi-Markov chain version further shows its superiority due to significantly better ($p < 0.05$) than others.

Table 6.2: T-tests of generative log-likelihoods (LL) of hypnograms on dynamical models. CI is confidence interval for the log likelihood difference in each case.

WNR	**H value**	**P value**	**CI**
LL(MM) - LL(SMM)	1	0.0086	[-18.5311, -2.7123]
LL(SMM) - LL(HMM)	1	0.0061	[3.1023, 18.5397]
WDL	**H value**	**P value**	**CI**
LL(MM) - LL(SMM)	1	0.0069	[-20.6407, -3.2936]
LL(SMM) - LL(HMM)	1	0.0217	[1.4640, 18.4472]
W5	**H value**	**P value**	**CI**
LL(MM) - LL(SMM)	1	2.8972e-11	[-53.9527, -29.7896]
LL(SMM) - LL(HMM)	1	3.3696e-13	[33.4582, 57.2735]

Initialization Evaluation: For deviation-based DTW clustering, Figure 6.5 shows the CDMC clusters (circles and triangles) coordinated by the parameters values of Weibull distribution (scale and shape) on wake stage in WNR dataset. The cluster assignments made by global weighted DTW clusters are better to separate patients in human sleep dataset into two clusters, in contrast with cluster boundaries made by standard DTW as the similarity metric.

During the below experimental procedure, model fit was significantly better for both global weighted DTW (gwDTW) clustering and stepwise deviated DTW (sdDTW) clustering as compared with standard DTW-driven clustering, as shown in Table 6.3. This shows that deviation-based DTW is superior to standard DTW as a similarity metric for initialization of CDMC clustering over human sleep data, as well as for standalone clustering over synthetic data as shown in section 2.4.

Experimental procedure, sleep data clustering:

```
*begin*

    D1, D2, D3 = W5, WNR, WDL datasets
```

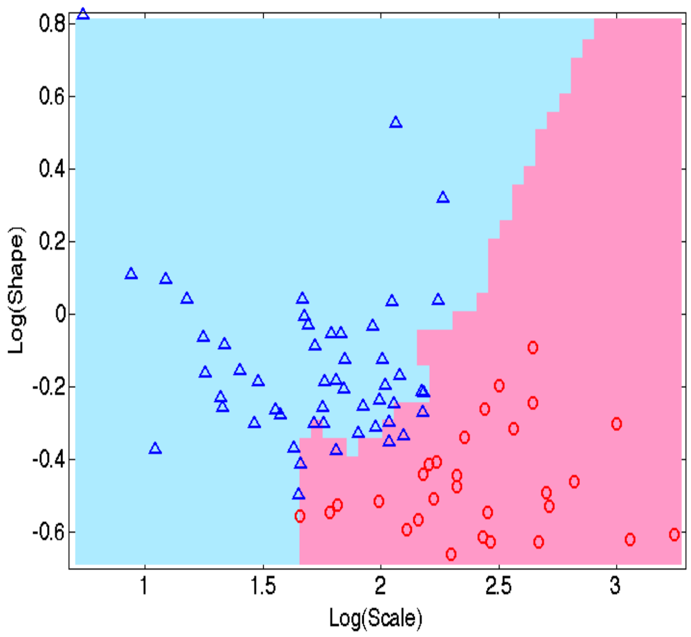


Figure 6.5: Visualization of clusters over human sleep dataset using gwDTW as similarity measure. Coordinates are Weibull shape and scale parameters for Wake stage. Red circles and blue triangles denote gwDTW clusters; background colors represent DTW clusters.

```
    m1, m2, m3 = $DTW, gwDTW, sdDTW$

    for $j :=$ 1 to 3

        for $k :=$ 1 to 3

            (M, nlogll(j, k, 1, ...244)) = $CDMC(D_j, m_k)$;

        end

        Perform Wilcoxon rank sum test on $nlogll(j, 1\dots3, 1\dots244)$

    end

*end*
```

Note:

- The W5, WNR, WDL datasets are as in section 6.1.1.
- DTW refers to clustering using standard DTW as the similarity metric; *gwDTW* and *sdDTW* refer to the deviation-based clustering techniques described in section 2.3.2.
- $CDMC(D_j, method)$ refers to CDMC clustering [34] with semi-Markov cluster models, using the given method for clustering initialization, and is assumed to return a set of dynamical models together with negative generative log likelihoods for all input sequences.

Table 6.3: Median negative log likelihoods of gwDTW, sdDTW, and standard DTW clusterings over WNR, WDL, and W5 human sleep datasets in section 6.1.1. Asterisks indicate Bonferroni-corrected significance of differences with standard DTW in Wilcoxon rank sum test ($p < 0.05$).

	DTW	**gwDTW**	**sdDTW**
WNR	150.7	148.2*	147.9*
WDL	159.4	157.9*	158.1*
W5	194.1	192.3*	191.9*

6.2 User Navigation Behavior

Nowadays, web usage mining has been defined as an essential technique of finding hidden knowledge from a web server log file, which records the interaction between website and user [118]. Web designers analyze the file and extract patterns from the trails formed by users during the navigation process to improve the web topology [119]. The prediction mining process in this section aims at predicting online user requests ahead of time and thus creating a robust web information service [31]. The main objectives in this section are displayed in three directions:

- Examine characteristics of page switches from one to another and switch durations.
- Present dynamical models for fitting web usage dataset well to do clustering task.
- Investigate effects of dynamical models in web data analysis about relationships between user habits and surfing histories.

6.2.1 Data Description

The MSNBC dataset (web user navigation behavior dataset) on a website comes from Internet Information Server (IIS) logs for msnbc.com on $September, 28, 1999$ (Pacific Standard Time) [20]. It has been applied to visualize navigation patterns in the light of model-based clustering [118].

Each sequence in the dataset corresponds to a webpage browsing history of a user during the day. Every event in a sequence corresponds to a user's request for a certain page. Requests are represented at the level of page category rather than at the level of uniform resource locator (URL). The categories consist of "frontpage", "news", "tech", and so on. There are totally 17 categories in this dataset. Each category is associated with an integer starting with "1" as listed in Table 6.4.

Table 6.4: Encodings of user navigation behavior categories.

Category	**Code**	**Category**	**Code**
Frontpage	1	Living	10
News	2	Business	11
Tech	3	Sports	12
Local	4	Summary	13
Opinion	5	BBS	14
On-air	6	Travel	15
Misc	7	msn-news	16
Weather	8	msn-sports	17
Health	9		

As shown in Table 6.5, there are as many as 1 million users in the dataset. For each user, there is a corresponding sequence to denote the user's visiting history in the server logs. About 90 percent of users have visits of length less than 10 requests. And the rest of users have more than 10 requests. There are 36 users who have more than 500 requests, which are considered as outliers, in the common sense that users could not stay at a website for a very long time.

Table 6.5: Statistics of MSNBC dataset.

Number of users	989818
Average number of visits per user	5.7
Average length of users' visiting history	4.74
Extra length of users' visiting history	36
The number of users' visits less than 10	887471
The number of users' visits more than 10	102347
The percentile of users' visits less than 10	89.66%
The percentile of users' visits more than 10	10.34%

Figure 6.6 shows the cumulative distribution of session lengths of 989818 users. For users with more than 10 requests, they take up only 10 percent of all users, approximately less than 10000 users. The observation implies that web users could be roughly divided into two types: users with short-term (session length less than 10) visits and users with long-term (session length more than 10) visits.

If the distribution of page transitions (from one category to another one, including self transition) is considered, Figure 6.7 illustrates that self-transitions of categories play an vital part in modeling web user navigation behavior, although there are a small portion of users (indicated by lighter squares) located far away from the diagonal line in the transition matrix.

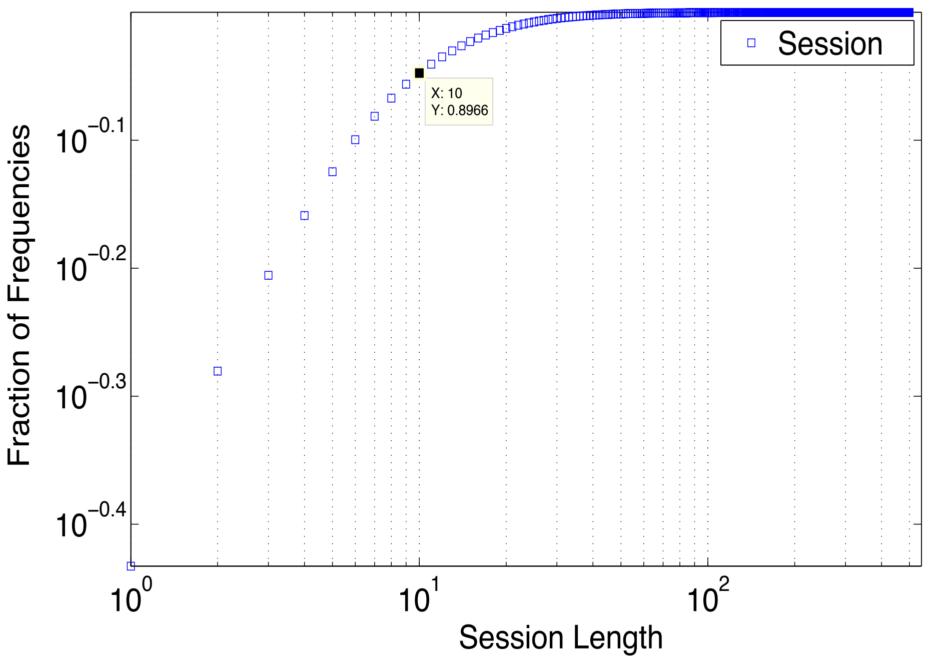


Figure 6.6: The cumulative distribution of session lengths over all users. As a cutoff point in session length, there are only 10 percent of users with more than 10 visits in the server logs. More than 89 percent of users belong to short-term visit type. In addition, there are 36 users whose session length is over 500 that are excluded in data analysis.

6.2.2 Dynamical Model Evaluation

In addition to the distribution of page transitions as shown in Figure 6.7, it is necessary to dig into distributions of page self-transitions. Since if Markov chain model [118] is taken into outlining dynamics of visiting histories on the server logs, it implies that the distribution of durations of a given category follows exponential distribution. Otherwise, it is wise to take different parametric model to depict the distribution of durations of a given category.

From Figure 6.8 to Figure 6.11, we show the distributions of short-term visitors and long-term visitors, divided by the length of sequences over all categories. Note that there are totally 17 categories and each category is encoded into a number as listed in Table 6.4. If the length of a sequence is less than or equal to 10, it is categorized into a short-term

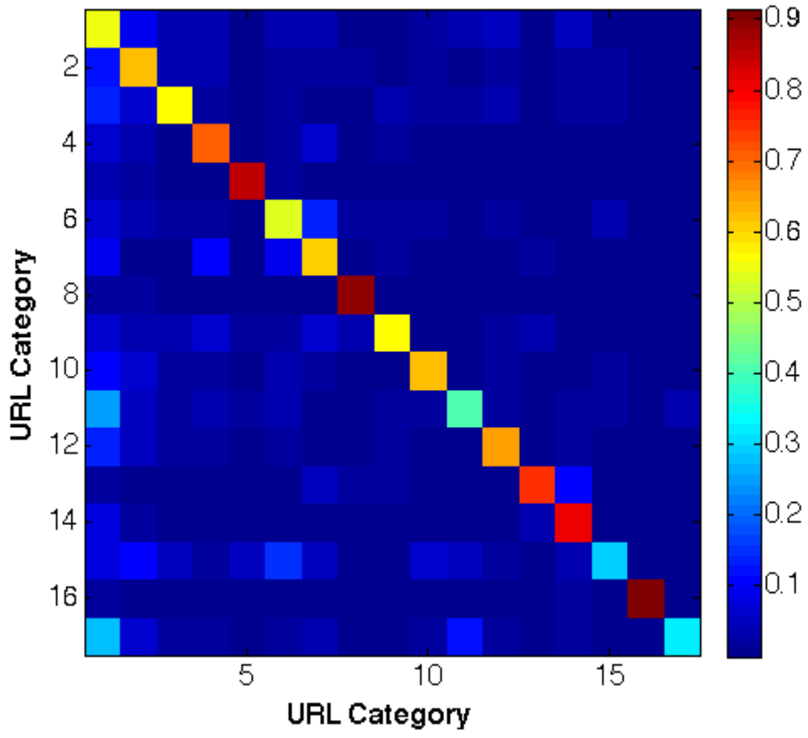


Figure 6.7: The distribution of web users browsing history. A lighter color indicates a higher probability of page switching transitions. Self-transitions within pages indicate browsing history from one page to another under the same category. It is a scaled graphical representation of a Markov transition matrix. Self-transitions within individual pages are much higher than any other transitions.

visitor; otherwise it is a long-term visitor.

For short-term visitors, there are many categories having good fitness for exponential distributions, such as categories 1, 2, 3, 4, 6, 9, 11, 12, 15, 16, 17. On the other hand, for Weibull distribution, all of categories have good fitness. Analyzed by fitting function formulations in section 3.3, Weibull distribution could be degenerated into exponential distribution when the shape parameter λ is equal to 1.

For long-term visitors, there are several categories such as 7, 9, 12, and 17 with good fitness for exponential distributions, while Weibull distribution performs much better than the exponential one. We show that for visiting patterns in the long-term visitors, Weibull fits are much better than exponential one to depict dynamical characteristics in user navigation patterns.

In a conclusion, the distributions of category durations in web user navigation behavior dataset possess Markov properties to some degree, such as good fitness of exponential distributions. Furthermore, it is better to use semi-Markov chains as dynamical models in the CDMC framework to cluster the dataset.

6.2.3 CDMC Framework Evaluation

As introduced in section 6.2.1, the MSNBC dataset was divided into two parts: short-term dataset and long-term dataset. The collective dynamical modeling clustering algorithm was used for clustering, with semi-Markov chains as dynamical models. Initial cluster guesses were provided by deviation-based DTW clustering as described in Algorithm 4, with either gwDTW (Equation 2.5) or sdDTW (Equation 2.6) as the similarity of sequences. Clustering driven by the standard DTW (Equation 2.4) distance metric was used as a basis for comparison.

Models Analysis: To fully examine dynamical models and various versions of web user navigation behavior described in section 6.2.1, three Markov models were respectively tested in pairs as dynamical models in the CDMC framework. Two clusters were built on the collection of short-term visits and long-term visits respectively due to their different characteristics. The t-tests of generative log-likelihoods (logarithmic values of probabilities of generating user requests given each Markov model) between each pair of models are shown in Table 6.6. Semi-Markov chain models with Weibull state durations perform significantly better than other models in terms of generative log-likelihoods on sequences of web user navigation behavior dataset. The confidence intervals on the difference of the log likelihoods means in Table 6.6 further establish the superiority of semi-Markov chains over Markov chains and hidden Markov models as dynamical models in CDMC.

Table 6.6: T-tests of generative log-likelihoods (LL) of web user navigation behavior dataset on dynamical models. CI is confidence interval for the log likelihood difference in each case.

Short-Term Dataset	**H value**	**P value**	**CI**
LL(MM) - LL(SMM)	1	0.0075	[-0.8948, -0.8715]
LL(SMM) - LL(HMM)	1	0.0059	[0.0165, 0.0586]
Long-Term Dataset	**H value**	**P value**	**CI**
LL(MM) - LL(SMM)	1	0.001	[-2.4373, -2.2522]
LL(SMM) - LL(HMM)	1	0.0138	[0.0204, 0.1796]

Therefore, CDMC with semi-Markov chains as dynamical models gives a higher probabilistic description of user navigation patterns, compared with the original CDMC with Markov chains or hidden Markov models as dynamical models.

Initialization Evaluation: Generative negative log likelihood was used to investigate the effect of initialization technique in the CDMC framework, with lower negative log-likelihood values (higher generative probabilities) indicating a better initial cluster guesses as input to the CDMC framework. The generative negative log-likelihood is defined as $-log(P(s|M))$ of a sequence, s, which is a measure of the probability that s, would be produced by a dynamical model M, built over a group of sequences in the dataset. The goal of clustering in CDMC was to minimize the generative negative log likelihood. Statistical difference of median negative log likelihoods with respect to different initialization techniques was tested by a Wilcoxon rank sum test.

Similar to the experimental procedure in section 6.1.3, model fit over web user navigation dataset was significantly better for both global weighted DTW (gwDTW) clustering and stepwise deviated DTW (sdDTW) clustering as compared with standard DTW-driven clustering, as shown in Table 6.7. Thus, deviation-based DTW is superior to standard DTW as a similarity metric for initialization of CDMC clustering over web user navigation data.

Table 6.7: Median negative log likelihoods of gwDTW, sdDTW, and standard DTW clusterings over short-term and long-term web user navigation behavior datasets in section 6.2.1. Asterisks indicate significance of differences with standard DTW in Wilcoxon rank sum test $(p < 0.05)$.

	DTW	**gwDTW**	**sdDTW**
Short-Term	5.360	5.322*	5.279*
Long-Term	27.76	26.38*	26.19*

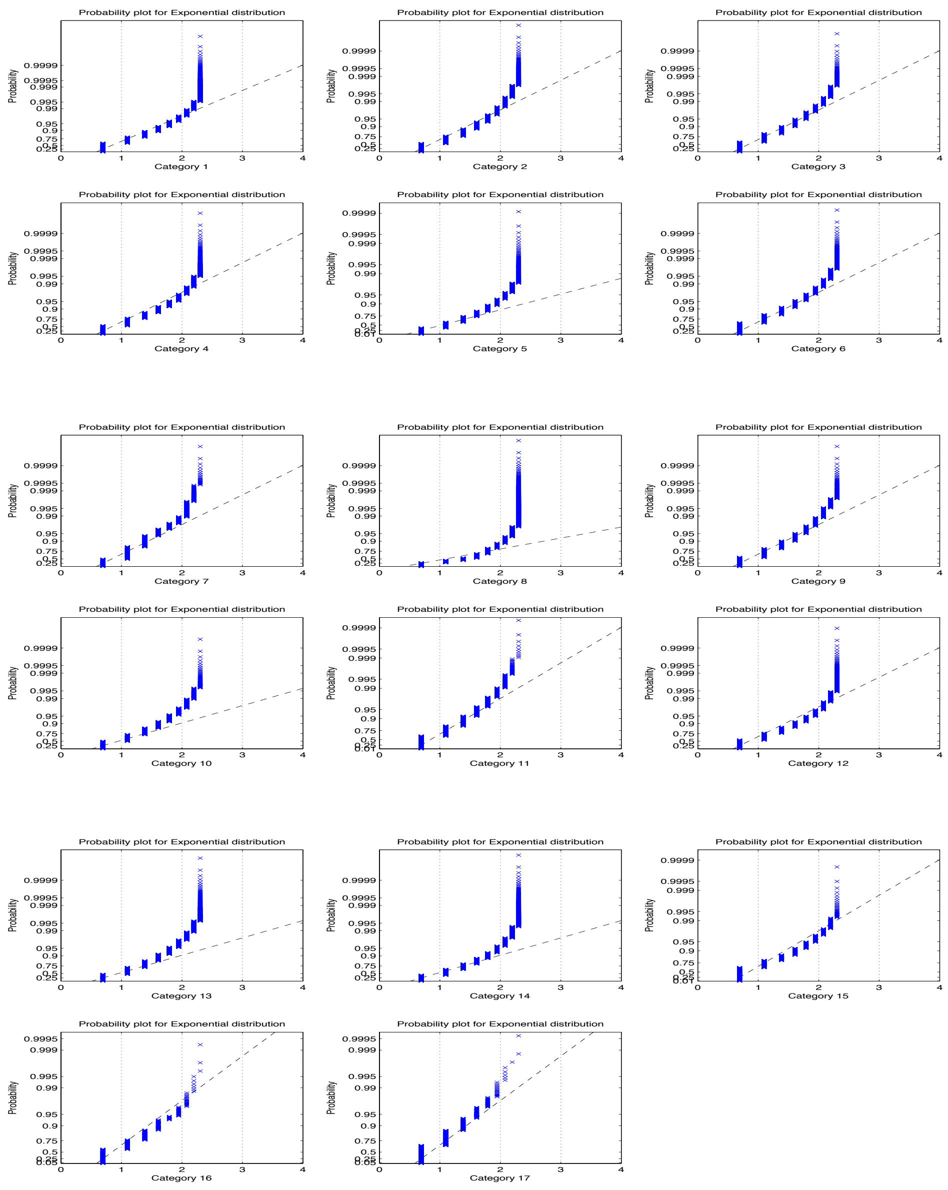


Figure 6.8: Exponential plots over all categories of duration distributions in short-term visiting history (sequences of length less than 10). The reference line (a dashed line) indicates a perfect exponential fit. Exponential fits are good to describe the short-term visitors (close to the reference line), since they only temporarily visit some website and then go away. Such patterns lead to the fact that the frequency of short durations of the given category would be much higher than the frequency of longer durations.

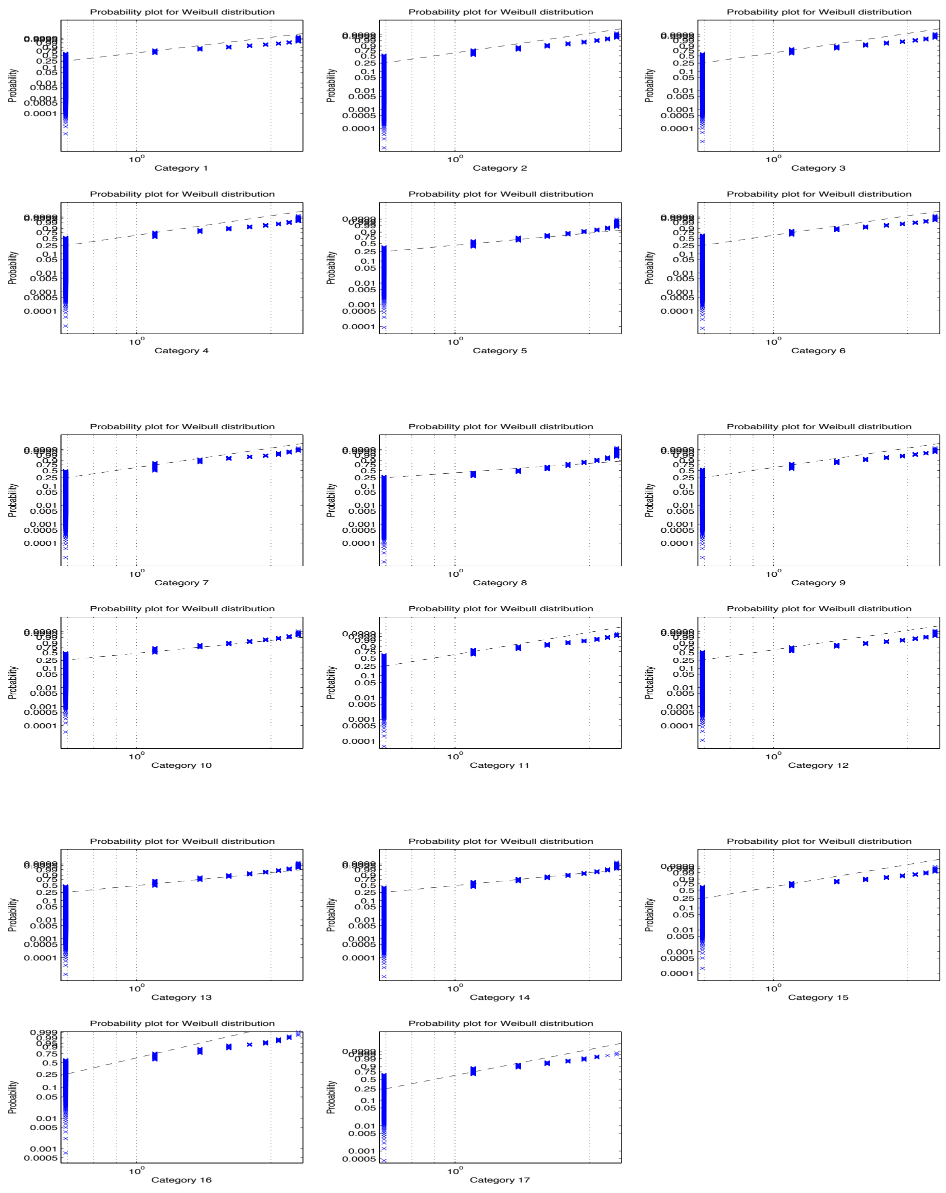


Figure 6.9: Weibull plots over all categories of duration distributions of short-term visiting history (sequences of length less than 10). The reference line indicates a perfect Weibull fit. The Weibull fits are better than exponential fit, since all distributions of categories should be much closer to the reference line, although there is a big tail at the beginning of the distribution of each category.

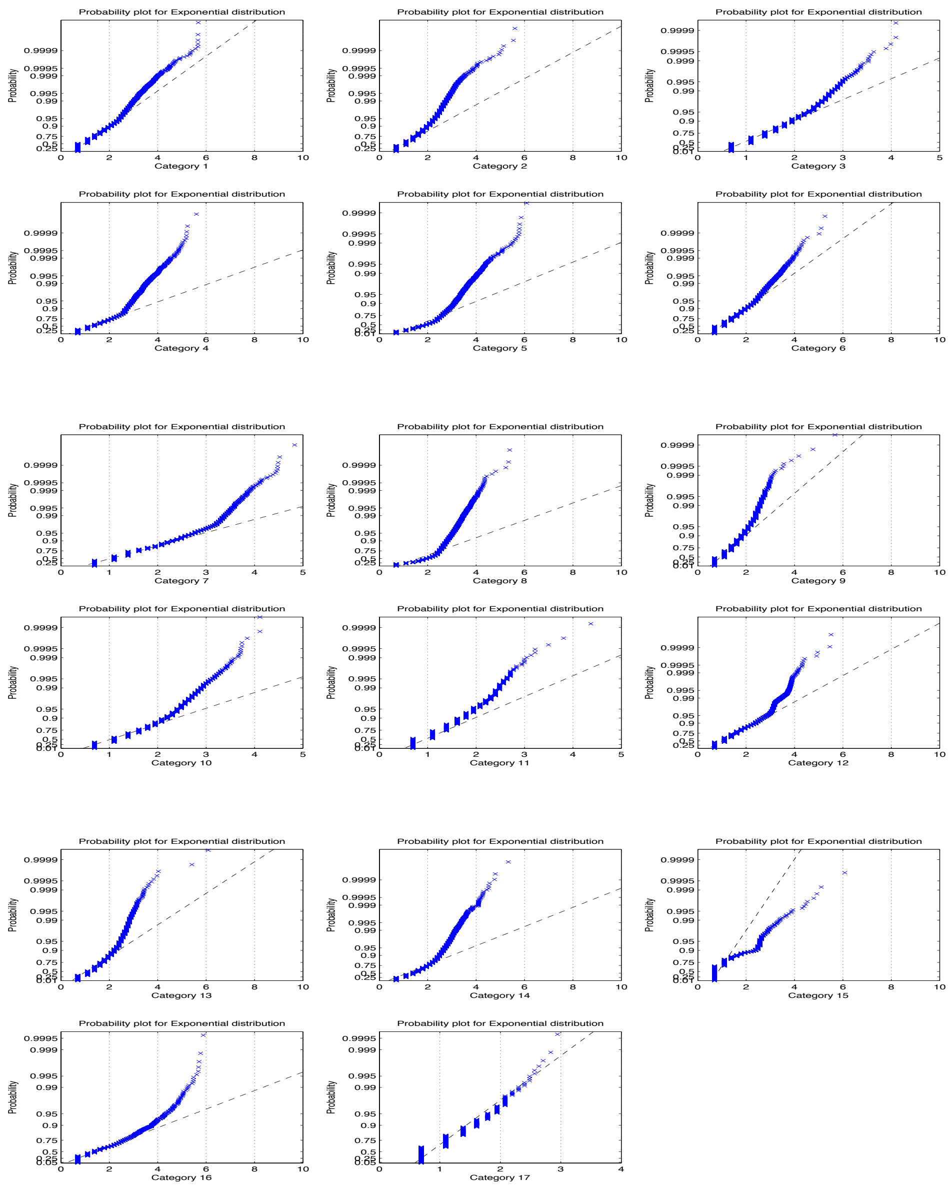


Figure 6.10: Exponential plots over all categories of duration distributions of long-term visiting history (sequences of length more than 10). The reference line indicates a perfect exponential fit. The exponential fits are worse than that in the situation for short-term visitors, since many distributions of categories are much far away from the duration distributions of categories.

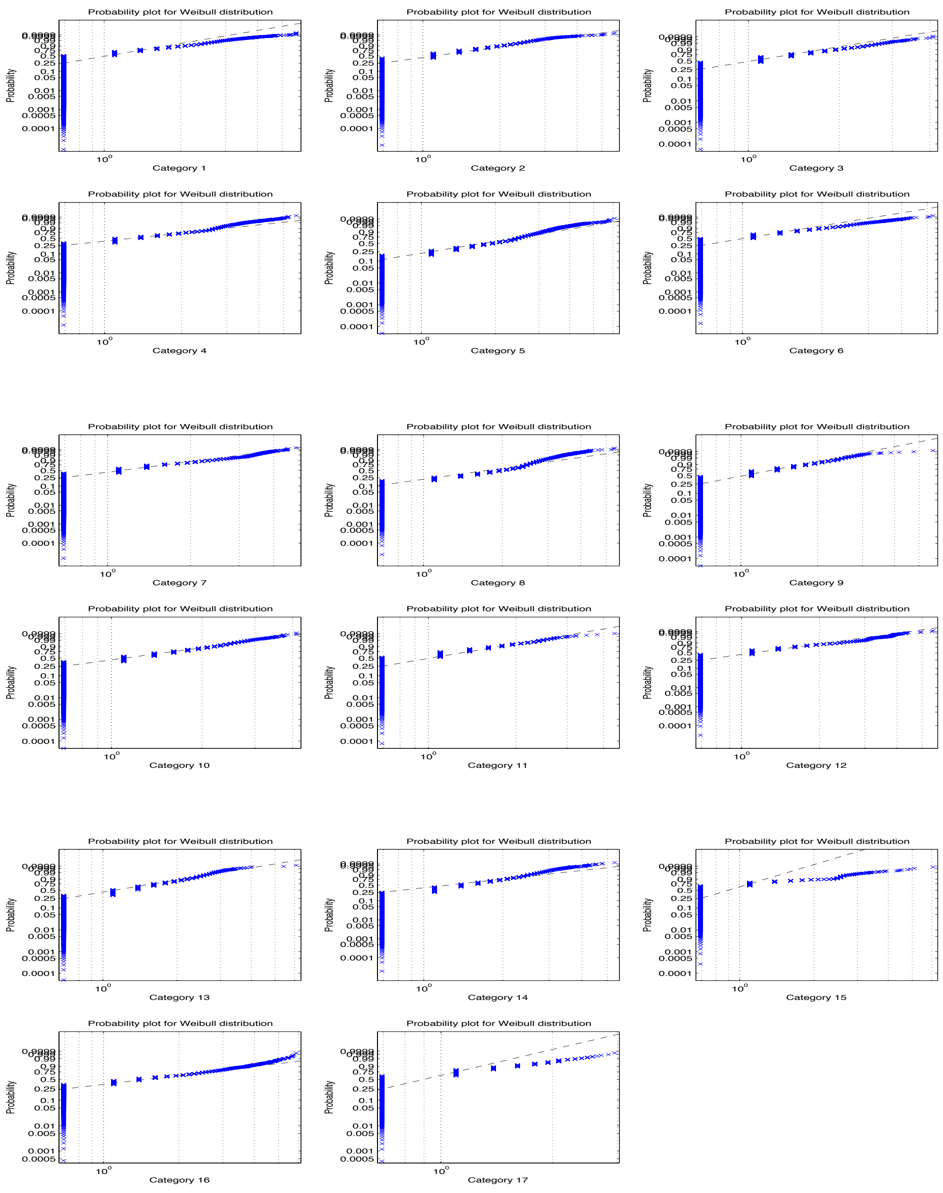


Figure 6.11: Weibull plots over all categories of duration distributions of long-term visiting history (sequences of length more than 10). The reference line indicates a perfect Weibull fit. Compared with the exponential fits, Weibull fits are much better than those for long-terms visitors, since there is a large impact of long duration of categories on the overall distribution of all categories.

Chapter 7

Conclusions of This Dissertation

The goal of this dissertation was to construct an automated framework for the discovery of dynamical patterns in time series with seldom occurrences of dynamical events. We investigated novel theoretical and practical aspects of the state-of-the-art sequence modeling technique, Collective Dynamical Modeling Clustering (CDMC) [34], and addressed the crucial need to scale up the original CDMC framework to process big data. The four major contributions of this dissertation can be summarized as follows.

First, we considered semi-Markov chains as models of discrete time series. Both state transitions and the durations of continuous bouts in each state are taken into account. A semi-Markov chain comprises an underlying Markov chain that models the temporal sequence of states but not the timing details, together with a separate statistical model of the bout durations in each state. The state bout durations are modeled explicitly by the Weibull parametric family of probability distributions. The Weibull semi-Markov chain model improves considerably on standard Markov chain models, which force geometrically distributed (discrete exponential) state bout durations for all states, contradicting known experimental observations. Our results provide more realistic dynamical modeling of sleep stage dynamics that can be expected to facilitate the discovery of interesting and useful dynamical patterns in human sleep data.

Second, we used semi-Markov chains as the CDMC dynamical models, as they better capture infrequent dynamical events in comparison with the more widely used Markov models. The experimental results over both synthetic data and real data (human sleep and user navigation behavior) confirm the validity of this statement. We proposed the use of distance-based dynamic time warping clustering for CDMC initialization, which led to significantly more accurate CDMC clustering results in experiments with labeled synthetic data than pseudorandom initialization does, as well as significantly faster CDMC convergence. We tested the adjusted Rand index as the clustering similarity metric that defines the CDMC stopping criterion, in order to correct for clustering agreements due to chance. However, it did not lead to significantly more accurate clusterings, while significantly increasing the number of iterations required for convergence as compared with the standard Rand index. Similar statements hold for the Normalized Mutual Information metric as compared with the standard Rand index. Therefore, the standard Rand index was the best choice of similarity metric for CDMC among these candidates.

Third, we proposed two versions of a modified dynamic time warping (DTW) approach for comparing discrete time series such as human sleep sequences: global weighted dynamic time warping (gwDTW) and stepwise deviated dynamic time warping (sdDTW). Both versions penalize deviations from the path of constant slope in the warping space, yielding the efficiency advantages of DTW approaches based on global constraints such as the Itakura parallelogram or the Sakoe-Chiba band, while better accounting for local deviations. gwDTW adds a deviation-based term to the standard DTW distance metric. sdDTW adds a deviation term into the local cost function that drives the DTW dynamic programming optimization itself, yielding an improved warping path together with a similarity metric. Both gwDTW and sdDTW lead to significantly better clustering results than DTW in a classification task over labeled synthetic Markov data, as well as in unsupervised clustering of human sleep data.

Lastly, we developed a distributed version of the collective dynamical modeling clustering algorithm. This distributed version scales up the original CDMC framework to process big data. The core components of our distributed approach include grouping incoming data instances and modeling over the aggregated dataset. We used Storm, an open source distributed real-time computation system, to support batch and distributed processing of data. A systematic evaluation of the proposed approach was carried out via experimentation with real-world data on human sleep and web user navigation behavior to establish the efficiency and scalability of our approach.

Chapter 8

Future Work

Future work may address further improvements to the initialization, dynamical modeling and distributed aspects of the collective dynamical modeling and clustering framework.

For the initialization component, dynamic time warping has been applied to the initialization of a different clustering technique based on Hidden Markov Models [51]. In the presence of big data, it would be worthwhile to investigate clustering time series using optimized dynamic time warping [120] and its integration into our current distributed CDMC framework. Another possible future work is to examine deviation-based constraints in dynamic time warping computation, compared with other constrained DTW work. For example, [121] mentioned a salient feature approach, which extracts features of the input sequences that are then used to define locally adaptive constraints on the warping path. The local constraints can affect the mean value of the warping band as a function of time, as well as its width. It would be desirable to pursue a performance comparison of the salient feature approach with that of our work.

For dynamical models, our work sheds light on investigating discrete time series analysis of state-based dynamics. For real-valued continuous time series, it is useful to consider approaches for converting them into discrete symbolic time series using dimensionality reduction techniques, more precisely feature extraction techniques. Future work could consider using Wavelet transform [109, 110] or Fourier transform [111] to efficiently detect signal components as user-defined states. An alternative approach is time series shapelets [112] which have been thought of as features for time series classification. It would be interesting to study the effect of using such features as part of our dynamical models during clustering in the CDMC framework. Another direction for future work is to investigate the use of semi-Markov models with hidden states, analogous to hidden Markov models. It is important to address whether temporal differences in discrete time series are adequately described by the transient (non-equilibrium) behavior of semi-Markov chains, or if it will instead be necessary to explicitly incorporate non-stationarity into the dynamical models.

For our distributed collective dynamical modeling clustering approach, we used Apache Storm as the distributed system platform. As optimizations of Hadoop [115] appear in new releases, it would be worthwhile to incorporate these optimizations in our framework. Another aspect worth exploring is that of estimating an appropriate value for the number of iterations of modeling and clustering in our framework – too large or too small a value may cause slow response or inadequate modeling and clustering results.

References

[1] Chotirat Ann Ratanamahatana and Eamonn Keogh. Making time-series classification more accurate using learned constraints. In *Proceedings of SIAM International Conference on Data Mining (SDM'04)*, pages 11–22. SIAM, 2004. 1, 23

[2] Wayne F Velicer and Joseph L Fava. Time series analysis. *J. Schinka & WF Velicer (Eds.), Research Methods in Psychology*, 2:581–606, 2003. 1

[3] Jessica Lin, Eamonn J. Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007. 1

[4] Martin T Hagan, Howard B Demuth, Mark H Beale, and Orlando De Jesús. *Neural network design*, volume 20. PWS Publishing Company, Boston, 1996. 1

[5] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011. 1

[6] C Stuart Daw, Charles Edward Andrew Finney, and Eugene R Tracy. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 74(2):915–930, 2003. 1

[7] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic represen-

tation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, pages 2–11. ACM, 2003. 1, 14

[8] Joseph W Burns, Leslie J Crofford, and Ronald D Chervin. Sleep stage dynamics in fibromyalgia patients and controls. *Sleep Medicine*, 9(6):689–696, 2008. 2, 41

[9] Matt T Bianchi, Sydney S Cash, Joseph Mietus, Chung-Kang Peng, and Robert Thomas. Obstructive sleep apnea alters sleep stage transition dynamics. *PloS One*, 5(6):e11356, 2010. 2, 6, 8, 102

[10] J. Chu-Shore, M. B. Westover, and M. T. Bianchi. Power law versus exponential state transition dynamics: application to sleep-wake architecture. *PloS One*, 5(12):e14204, 2010. 2, 41, 43, 44, 102

[11] A. Kishi, Z. R. Struzik, B. H. Natelson, F. Togo, and Y. Yamamoto. Dynamics of sleep stage transitions in healthy humans and patients with chronic fatigue syndrome. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 294(6):R1980–R1987, 2008. 2, 44, 58, 102

[12] Bob Kemp and HA Kamphuisen. Simulation of human hypnograms using a Markov chain model. *Sleep*, 9(3):405–414, 1986. 2

[13] Mathias Basner and Uwe Siebert. *Markov state transition models for the prediction of changes in sleep structure induced by aircraft noise*. Deutsches Zentrum für Luft-und Raumfahrt eV DLR, Bibliotheks-und Informationswesen, 2006. 2

[14] W. W. Zung, T. H. Naylor, D. T. Gianturco, and W. P. Wilson. Computer simulation of sleep EEG patterns with a Markov chain model. *Recent Advances in Biological Psychiatry*, 8:335–355, 1965. 2, 42

[15] Chiying Wang, Sergio A. Alvarez, Carolina Ruiz, and Majaz Moonis. Computational modeling of sleep stage dynamics using Weibull semi-Markov chains. *Sixth International Conference on Health Informatics (HEALTHINF 2013)*, pages 122–130, February 2013. 2, 12, 40, 43, 50, 57

[16] JW Kim, J. S. Lee, PA Robinson, and D. U. Jeong. Markov analysis of sleep dynamics. *Physical Review Letters*, 102(17):178104–178104, 2009. 2, 43

[17] C-C Lo, LA Nunes Amaral, Shlomo Havlin, P Ch Ivanov, Thomas Penzel, J-H Peter, and H Eugene Stanley. Dynamics of sleep-wake transitions during sleep. *Europhysics Letters*, 57(5):625–631, 2002. 3, 4

[18] Corrado Cavallero, Piercarla Cicogna, Vincenzo Natale, Miranda Occhionero, et al. Slow wave sleep dreaming. *Sleep*, 15(6):562–566, 1992. 3, 55

[19] Doris Moser, Peter Anderer, Georg Gruber, Silvia Parapatics, Erna Loretz, Marion Boeck, Gerhard Kloesch, Esther Heller, Andrea Schmidt, Heidi Danker-Hopfe, et al. Sleep classification according to AASM and Rechtschaffen & Kales: effects on sleep scoring parameters. *Sleep*, 32(2):139–149, 2009. 3, 55

[20] David Heckerman. Anonymous Web Data Data Set, September 1999. `http://archive.ics.uci.edu/ml/datasets/msnbc.com+anonymous+web+data`. 2, 4, 112

[21] Habel Kurian. A Markov model for web request prediction. Master's thesis, Department of Computing and Information Sciences, Kansas State University, 2008. 3

[22] Jiu Jun Chen, Ji Gao, Jun Hu, and Bei Shui Liao. Dynamic mining for web navigation patterns based on Markov model. In *Computational and Information Science, First International Symposium, CIS 2004*, pages 806–811, 2004. 3

[23] Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice Hall signal processing series. Prentice Hall, 1993. 3

[24] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 3, 41, 42, 43

[25] Himanshu Joshi, Paul S Rosenbloom, and Volkan Ustun. Isolated word recognition in the Sigma cognitive architecture. *Biologically Inspired Cognitive Architectures*, 10:1–9, 2014. 3, 5

[26] Thomas P Minka. From hidden Markov models to linear dynamical systems. Technical report, Vision and Modeling Group of Media Lab, MIT, 1999. 3, 41

[27] Bertrand Mesot and David Barber. Switching linear dynamical systems for noise robust speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(6):1850–1858, 2007. 4, 42

[28] Allan Rechtschaffen and Anthony Kales. *A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects*. US Department of Health, Education, and Welfare Public Health Service - NIH/NIND, 1968. 4

[29] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. 6

[30] Marco Ramoni, Paola Sebastiani, and Paul R. Cohen. Bayesian clustering by dynamics. *Machine Learning*, 47(1):91–121, 2002. 7

[31] Igor V. Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Model-based clustering and visualization of navigation patterns on a web site. *Data Mining and Knowledge Discovery*, 7(4):399–424, 2003. 7, 62, 96, 111

[32] Padhraic Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems*, pages 648–654. Citeseer, 1996. 7

[33] Julia Sivriver, Naomi Habib, and Nir Friedman. An integrative clustering and modeling algorithm for dynamical gene expression data. *Bioinformatics*, 27(13):392–400, 2011. 8, 62

[34] Sergio A. Alvarez and Carolina Ruiz. Collective probabilistic dynamical modeling of sleep stage transitions. *Sixth International Conference on Bio-inspired Systems and Signal Processing (BIOSIGNALS 2013)*, pages 209–214, February 2013. 8, 9, 10, 32, 62, 78, 102, 111, 123

[35] Chiying Wang, Sergio A. Alvarez, Carolina Ruiz, and Majaz Moonis. Deviation-based dynamic time warping for clustering human sleep. *Ninth International Conference on Bio-inspired Systems and Signal Processing (BIOSIGNALS 2016)*, February 2016. 12, 86

[36] Apache Software Foundation. Apache Storm, December 2013. `http://storm.incubator.apache.org/`. 13, 80

[37] Eamonn J. Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003. 14

[38] Dominique M Hanssens, Leonard J Parsons, and Randall L Schultz. *Market response models: Econometric and time series analysis*, volume 12. Springer Science & Business Media, 2003. 14

[39] Kazuhiro Kohara, Tsutomu Ishikawa, Yoshimi Fukuhara, and Yukihiro Nakamura. Stock price prediction using prior knowledge and neural networks. *Intelligent Systems in Accounting, Finance and Management*, 6(1):11–22, 1997. 14

[40] Brainl A Rhatigan, Kenneth C Mylrea, Edward Lonsdale, and Lawrence Z Stern. Investigation of sounds produced by healthy and diseased human muscular contraction. *IEEE Transactions on Biomedical Engineering*, 33(10):967–971, 1986. 14

[41] Themis Palpanas, Michail Vlachos, Eamonn J. Keogh, and Dimitrios Gunopulos. Streaming time series summarization using user-defined amnesic functions. *IEEE Transactions on Knowledge and Data Engineering*, 20(7):992–1006, 2008. 14

[42] Ruey-Chyn Tsaur, Jia-Chi O Yang, and Hsiao-Fan Wang. Fuzzy relation analysis in fuzzy time series model. *Computers & Mathematics with Applications*, 49(4):539–548, 2005. 14

[43] Rui Xu and Donald C. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005. 14, 33

[44] T Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, 2005. 14, 15

[45] K. Chidananda Gowda and Edwin Diday. Symbolic clustering using a new similarity measure. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):368–378, 1992. 14

[46] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. Technical report, Michigan State Universiy, 2006. 15

[47] Eamonn J. Keogh and Chotirat (Ann) Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005. 15

[48] Eamonn J Keogh and Michael J Pazzani. Derivative dynamic time warping. In

Proceedings of SIAM International Conference on Data Mining (SDM'01), pages 1–12. SIAM, 2001. 15

[49] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *AAAI-94 Workshop on Knowledge Discovery in Databases*, volume 10, pages 359–370, 1994. 16, 19

[50] Jesin Zakaria, Abdullah Mueen, and Eamonn J. Keogh. Clustering time series using unsupervised-shapelets. In *12th IEEE International Conference on Data Mining, ICDM 2012*, pages 785–794. IEEE, 2012. 16

[51] Tim Oates, Laura Firoiu, and Paul R. Cohen. Using dynamic time warping to bootstrap HMM-based clustering of time series. In *Sequence Learning - Paradigms, Algorithms, and Applications*, pages 35–52. Springer, 2001. 16, 20, 102, 126

[52] EG Caiani, A Porta, G Baselli, M Turiel, S Muzzupappa, F Pieruzzi, C Crema, A Malliani, and S Cerutti. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. *IEEE Computers in Cardiology*, 25:73–76, 1998. 16

[53] John Aach and George M. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001. 16

[54] Yunyue Zhu and Dennis Shasha. Warping indexes with envelope transforms for query by humming. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 181–192. ACM, 2003. 16

[55] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978. 16, 19, 20, 21

[56] Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):67–72, 1975. 16, 19, 21

[57] Meinard Müller. Dynamic time warping. *Information Retrieval for Music and Motion*, pages 69–84, 2007. 16, 18, 102

[58] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007. 18, 20

[59] Chiying Wang, Sergio A. Alvarez, Carolina Ruiz, and Majaz Moonis. Semi-Markov modeling-clustering of human sleep with efficient initialization and stopping. *Seventh International Conference on Bio-inspired Systems and Signal Processing (BIOSIGNALS 2014)*, pages 61–68, March 2014. 20, 96, 102

[60] David Clifford, Glenn Stone, Ivan Montoliu, Serge Rezzi, François-Pierre Martin, Philippe Guy, Stephen Bruce, and Sunil Kochhar. Alignment using variable penalty dynamic time warping. *Analytical Chemistry*, 81(3):1000–1007, 2009. 22

[61] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining (SDM'05)*, pages 506–510. SIAM, 2005. 23

[62] Robert Goodell Brown. *Smoothing, forecasting and prediction of discrete time series*. Prentice Hall, 1962. 40

[63] Lawrence R Rabiner, Aaron E Rosenberg, and Stephen E Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *The Journal of the Acoustical Society of America*, 63(S1):S79–S79, 1978. 40

[64] Norbert Wiener. *Extrapolation, interpolation, and smoothing of stationary time series*, volume 2. MIT press Cambridge, MA, 1949. 40

[65] Tsu T Soong. *Fundamentals of Probability and Statistics for Engineers*. John Wiley & Sons, 2004. 41

[66] Blakeley B McShane, Raymond J Galante, Shane T Jensen, Nirinjini Naidoo, Allan I Pack, and Abraham Wyner. Characterization of the bout durations of sleep and wakefulness. *Journal of Neuroscience Methods*, 193(2):321–333, 2010. 41

[67] Iain L MacDonald and Walter Zucchini. *Hidden Markov and other models for discrete-valued time series*, volume 110. CRC Press, 1997. 41

[68] Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden Markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986. 41

[69] Georgios Sigletos, Georgios Paliouras, and Vangelis Karkaletsis. Role identification from free text using hidden markov models. In *Methods and Applications of Artificial Intelligence*, pages 167–178. Springer, 2002. 41

[70] Ahmet D Sahin and Zekai Sen. First-order Markov chain approach to wind speed modelling. *Journal of Wind Engineering and Industrial Aerodynamics*, 89(3):263–269, 2001. 42

[71] Mark CK Yang and Carolyn J Hursch. The use of a semi-Markov model for describing sleep patterns. *Biometrics*, 29(4):667–676, 1973. 43

[72] Rameshwar D Gupta and Debasis Kundu. Exponentiated exponential family: an alternative to gamma and Weibull distributions. *Biometrical Journal*, 43(1):117–130, 2001. 44

[73] Vincenzo Guerriero. Power law distribution: Method of multi-scale inferential statistics. *Journal of Modern Mathematics Frontier*, 1(1):21–28, 2012. 45

[74] Athanasios Papoulis and S Unnikrishna Pillai. Probability, random variables, and stochastic processes. *McGraw-Hill*, 1985. 46

[75] Zdravko I Botev, Joseph F Grotowski, Dirk P Kroese, et al. Kernel density estimation via diffusion. *Annals of Statistics*, 38(5):2916–2957, 2010. 46

[76] Asha Seth Kapadia, Wenyaw Chan, and Lemuel A Moyé. *Mathematical statistics with applications*. CRC Press, 2005. 47

[77] Johann Pfanzagl. *Parametric statistical theory*. Walter de Gruyter, 1994. 48

[78] Shun-Zheng Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243, 2010. 49

[79] Richard B Berry, Rita Brooks, Charlene E Gamaldo, SM Hardling, CL Marcus, and BV Vaughn. The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications. *American Academy of Sleep Medicine*, 2012. 55, 103

[80] John G Kemeny and James Laurie Snell. *Finite Markov chains*, volume 356. D. van Nostrand Company, Inc. Princeton, New Jersey, 1960. 55

[81] Igor V Cadez, Scott Gaffney, and Padhraic Smyth. A general probabilistic framework for clustering individuals and objects. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 140–149. ACM, 2000. 62

[82] Shi Zhong and Joydeep Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1037, 2003. 62

[83] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. 62, 63, 69

[84] CF Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11(1):95–103, 1983. 63, 69

[85] Bernard Delyon, Marc Lavielle, and Eric Moulines. Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics*, 27(1):94–128, 1999. 63

[86] Tom M. Mitchell. *Machine Learning*. McGraw Hill series in computer science. McGraw-Hill, 1997. 63, 64

[87] Jeff A Bilmes et al. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International Computer Science Institute*, 4(510):126, 1998. 64

[88] Florin Vaida. Parameter convergence for EM and MM algorithms. *Statistica Sinica*, 15(3):831–840, 2005. 70

[89] Silke Wagner and Dorothea Wagner. Comparing clusterings: an overview. Technical report, Universität Karlsruhe, Fakultät für Informatik, 2007. 70

[90] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press, 2008. 71

[91] Leslie C Morey and Alan Agresti. The measurement of classification agreement: An adjustment to the Rand statistic for chance agreement. *Educational and Psychological Measurement*, 44(1):33–37, 1984. 71

[92] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. 71

[93] Marina Meilă. Comparing clusterings–an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007. 72

[94] Xuan Vinh Nguyen, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009*, pages 1073–1080. ACM, 2009. 72

[95] Ana L. N. Fred and Anil K. Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume 2, pages 128–136. IEEE, 2003. 72

[96] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001. 72

[97] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107, 2014. 78

[98] Alexandros Labrinidis and HV Jagadish. Challenges and opportunities with big data. *Proceedings of the VLDB Endowment*, 5(12):2032–2033, 2012. 78

[99] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela H Byers. Big data: The next frontier for innovation, competition, and productivity. Technical report, McKinsey Global Institute, 2011. 78

[100] Wei Fan and Albert Bifet. Mining big data: current status, and forecast to the future. *ACM SIGKDD Explorations Newsletter*, 14(2):1–5, 2013. 78

[101] Tilmann Rabl, Sergio Gómez-Villamor, Mohammad Sadoghi, Victor Muntés-Mulero, Hans-Arno Jacobsen, and Serge Mankovskii. Solving big data challenges for enterprise application performance management. *Proceedings of the VLDB Endowment*, 5(12):1724–1735, 2012. 78

[102] Charu C Aggarwal. On effective classification of strings with wavelets. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 163–172. ACM, 2002. 79

[103] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1987. 79

[104] Nadia A. Chuzhanova, Antonia J. Jones, and Steve Margetts. Feature selection for genetic sequence classification. *Bioinformatics*, 14(2):139–143, 1998. 79

[105] Ssu-Hua Huang, Ru-Sheng Liu, Chien-Yu Chen, Ya-Ting Chao, and Shu-Yuan Chen. Prediction of outer membrane proteins by support vector machines using combinations of gapped amino acid pair compositions. In *Fifth IEEE International Symposium on Bioinformatic and Bioengineering (BIBE 2005)*, pages 113–120. IEEE, 2005. 79

[106] Daniel Kudenko and Haym Hirsh. Feature generation for sequence categorization. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 733–738. American Association for Artificial Intelligence, 1998. 79

[107] Neal Lesh, Mohammed J Zaki, and Mitsunori Ogihara. Mining features for sequence classification. In *Proceedings of the 5th ACM SIGKDD International Con-*

ference on Knowledge Discovery and Data Mining, pages 342–346. ACM, 1999. 79

[108] Xiaonan Ji, James Bailey, and Guozhu Dong. Mining minimal distinguishing subsequence patterns with gap constraints. *Knowledge and Information Systems*, 11(3):259–286, 2007. 79

[109] Wei-Ying Ma and B. S. Manjunath. A comparison of wavelet transform features for texture image annotation. In *Proceedings 1995 International Conference on Image Processing*, volume 2, pages 256–259. IEEE, 1995. 79, 127

[110] Aleš Procházka, Jaromír Kukal, and Oldřich Vyšata. Wavelet transform use for feature extraction and EEG signal segments classification. In *Proceedings of the Third International Symposium on Communications, Control and Signal Processing*, pages 719–722. ISCCSP, 2008. 79, 127

[111] Gang Liu, Di Wu, Hong-Gang Zhang, and Jun Guo. A new feature extraction method based on Fourier transform in handwriting digits recognition. In *Proceedings 2002 International Conference on Machine Learning and Cybernetics*, volume 3, pages 1227–1231. IEEE, 2002. 79, 127

[112] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956. ACM, 2009. 79, 127

[113] Francis W. Usher, Chiying Wang, Sergio A. Alvarez, Carolina Ruiz, and Majaz Moonis. Machine learning of human sleep patterns based on stage bout durations. *Fifth International Conference on Health Informatics (HEALTHINF 2012)*, pages 71–80, February 2012. 79

[114] Paul Zikopoulos and Chris Eaton. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 2011. 80

[115] Apache Software Foundation. Apache Hadoop, September 2007. `http://hadoop.apache.org/`. 80, 127

[116] Amro Khasawneh, Sergio A. Alvarez, Carolina Ruiz, Shivin Misra, and Majaz Moonis. EEG and ECG characteristics of human sleep composition types. *Fourth International Conference on Health Informatics (HEALTHINF 2011)*, pages 97–106, January 2011. Best Paper Award. 101, 103

[117] Derk-Jan Dijk and Steven W. Lockley. Invited Review: Integration of human sleep-wake regulation and circadian rhythmicity. *Journal of Applied Physiology*, 92(2):852–862, 2002. 102

[118] Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Visualization of navigation patterns on a web site using model-based clustering. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 280–284. ACM, 2000. 111, 112, 114

[119] V. Valli Mayil. Web navigation path pattern prediction using first order Markov model and depth first evaluation. *International Journal of Computer Applications (0975-8887)*, 45(16), 2012. 111

[120] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Pro-*

ceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 262–270. ACM, 2012. 126

[121] K Selçuk Candan, Rosaria Rossini, Xiaolan Wang, and Maria Luisa Sapino. sDTW: computing DTW distances using locally relevant constraints based on salient feature alignments. *Proceedings of the VLDB Endowment*, 5(11):1519–1530, 2012. 126