5-8-2020

Clustering of Time Series Data: Measures, Methods, and Applications

Ruizhe Ma

CLUSTERING OF TIME SERIES DATA: MEASURES, METHODS, AND APPLICATIONS

by

RUIZHE MA

Under the Direction of Rafal Angryk, PhD

ABSTRACT

Clustering is an essential branch of data mining and statistical analysis that could help us explore the distribution of data and extract knowledge. With the broad accumulation and application of time series data, the study of its clustering is a natural extension of existing unsupervised learning heuristics. We discuss the components which configure the clustering of time series data, specifically, the similarity measure, the clustering heuristic, the evaluation of cluster quality, and the applications of said heuristics. Being the groundwork for the task of data analysis, we propose a scalable and efficient time series similarity measure: segmented-Dynamic Time Warping. For time series clustering, we formulate the Distance Density Clustering heuristic, a deterministic clustering algorithm that adopts concepts from both density and distance separation. In addition, we explored the characteristics and discussed the limitations of existing cluster evaluation methods. Finally, all components lead to the goal of real-world applications.

INDEX WORDS: Data mining, Time series, Clustering

CLUSTERING OF TIME SERIES DATA: MEASURES, METHODS, AND APPLICATIONS

by

RUIZHE MA

A Dissertation Submitted in Partial Fulfillment for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2020

CLUSTERING OF TIME SERIES DATA: MEASURES, METHODS, AND APPLICATIONS

by

RUIZHE MA

Committee Chair:	Rafal Angryk
Committee:	Zhipeng Cai
	Petrus Martens
	Pete Riley
	Yanqing Zhang

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

May 2020

DEDICATION

This work is dedicated to my husband Bing, your understanding, patience, and support gave me the strength to complete this work. To my parents and brother, your love and encouragement have been invaluable to me all my life.

ACKNOWLEDGEMENTS

The work presented in this dissertation was made possible with the support of many people. First, I wish to express my sincere gratitude to the best advisor one could ask for, Dr. Rafal Angryk. Dr. Angryk's confidence in my abilities, his encouragement, supervision, and the push he provided helped me tremendously in the completion of this goal.

I would also like to thank each of my committee members – Dr. Zhipeng Cai, Dr. Piet Martens, Dr. Pete Riley, and Dr. Yanqing Zhang, for their continual support and guidance in my graduate studies.

I appreciate all the discussions, feedback, assistance, and friendship from the members of our lab over the years–Ahmet, Annie, Azim, Berkay, Dustin, Hamdi, Manolis, Max, Mike, Nikhil, Sajitha, Shailesh, Soukaina, Soumitra, Sunitha, Surabhi, Sushant, Vijay, Xumin, and Yang. Finally, to all my friends and family, thank you, I am honored to have you all in my life.

FUNDING ACKNOWLEDGEMENT

This project has been supported in part by funding from the Division of Advanced Cyberinfrastructure within the Directorate for Computer and Information Science and Engineering, the Division of Astronomical Sciences within the Directorate for Mathematical and Physical Sciences, and the Division of Atmospheric and Geospace Sciences within the Directorate for Geosciences, under NSF awards #1443061, #1812964, #1936361 and #1931555. It was also supported in part by funding from NASA, through the Heliophysics' Living With a Star Science Program, under NASA award #NNX15AF39G, as well as through the direct contract from Space Radiation Analysis Group (SRAG). In addition to that, the work has been in part sponsored by state funding from Georgia State University's Second Century Initiative, and Next Generation Program.

TABLE OF CONTENTS

LIST OF TABLES ix

LIST OF FIGURES xv

1 INTRODUCTION 1

1.1 Motivation 3

1.2 Challenges 3

1.3 Outline 5

2 BACKGROUND 6

2.1 Similarity Measure 6

2.1.1 DTW Improvement Techniques 12

2.1.2 Normalization 16

2.2 Clustering 19

2.2.1 Hierarchical Clustering 21

2.2.2 K-means Clustering 24

2.2.3 DBSCAN 25

2.2.4 OPTICS 26

2.3 Cluster Representation 29

2.4 Clustering Evaluation 31

2.4.1 Internal Indices 32

2.4.2 External Indices 33

2.4.3 Dendrogram Evaluation 34

2.5 Datasets 36

2.5.1 UCR Archive 36

2.5.2 ICME Data . . . 37
2.5.3 Solar Flare Data . . . 40
3 SEGMENTED DYNAMIC TIME WARPING (SEGDTW) . . . 46
3.1 Feature Selection . . . 48
3.2 Feature Pairing . . . 49
3.3 segDTW Performance . . . 53
3.3.1 Feature Type . . . 54
3.3.2 Peak Selection Parameter Effect on Performance . . . 55
3.3.3 segDTW Efficiency . . . 57
3.3.4 segDTW Application . . . 58
4 DISTANCE DENSITY CLUSTERING . . . 61
4.1 Clustering Initialization . . . 61
4.2 Cluster Update . . . 64
4.3 DDC Performance . . . 66
5 CLUSTER QUALITY EVALUATION . . . 68
5.1 Data Analysis . . . 69
5.2 Clustering Validation with Internal and External Indices . . . 71
5.3 Cluster Variance Evaluation . . . 73
6 EXPERIMENTS AND APPLICATIONS . . . 76
6.1 Hierarchical Structure Comparisons . . . 76
6.1.1 Internal Evaluation . . . 77
6.1.2 External Evaluation . . . 81
6.2 Decision Trees . . . 83
6.3 ICME Decision Trees . . . 84
6.3.1 DDC Decision Tree . . . 85
6.3.2 AGNES Decision Tree . . . 88
6.4 Solar Flare Decision Tree . . . 92

6.5 Time Series Cluster Profile 95
7 CONCLUSION AND FUTURE WORK 102
7.1 Conclusion 102
7.2 Future Work 103
REFERENCES 104

List of Tables

Table 2.1 Confusion Matrix 33

Table 2.2 ICME Parameters 39

Table 2.3 Active Region Parameters 42

Table 2.4 Solar flare datasets with prior $\in \{12, 24\}$ and span $\in \{6, 12, 24\}$ . . 42

Table 2.5 Labeled flare instances as per flare class for the five partitions of the SWAN-SF benchmark dataset. Also shown is the approximate class imbalance ratio between labels corresponding to M- and X-class flares and all the other labels. 45

Table 3.1 Time and Accuracy for DTW, segDTW, and the Combination of segDTW with the Itakura Window 59

Table 4.1 Similarity Matrix for Five Time Series 62

Table 6.1 Cophenetics Values for Variations of OPTICS and AGNES. . . . 79

Table 6.2 DDC categorical result table for building decision trees. 87

Table 6.3 Average Decision Tree Accuracy 90

Table 6.4 Leaf Node Purity and Accuracy for Solar Flare Decision Tree . . 95

Table 6.5 Nine Parameters Selected for Solar Pre-Flare Time Series Evaluation 96

List of Figures

Figure 1.1 (a) The discrete averaged time series distributed in the Euclidean space, (b) original time series for parameter A, and (c) original time series for parameter B. 2

Figure 2.1 The mappings of lock-step and elastic similarity measures: (a) Euclidean distance and (b) Dynamic Time Warping. 8

Figure 2.2 The Dynamic Time Warping warping path between time series Q and C. 9

Figure 2.3 Violation of the three basic conditions for DTW algorithm: (a) the boundary condition, (b) the monotonicity condition and (c) the continuity condition 10

Figure 2.4 DTW windows: (a) Sakoe-Chiba Band and (b) Itakura Parallelogram, only the colored cells will be computed, thus improving efficiency. Since the warping path is constrained, pathological warpings can be prevented to a certain degree. 13

Figure 2.5 Step Patterns: (a) *symmetric1* is the basic step pattern, (b) *symmetric2* favors the diagonal warping path, (c) *asymmetric* limits time expansion to a factor of two, and (d) *rabinerJuangStepPattern* has attributes local continuity constraint type, slope weighting, and the state of smoothed or not. 14

Figure 2.6 Offset translation shifts the range of time series to focus on the shape similarities. (a) Shows the original time series, (b) shows the normalized result after offset translation is applied. 17

Figure 2.7 Amplitude scaling changes the strength of the signal (time series) to make shape similarities easier to identify. (a) Shows the original time series, (b) shows the time series after amplitude removal. . . 18

Figure 2.8 Trend removal removes the patterns, or trends, from time series data in the hope of focusing on the value changes without the long-term trend distraction. (a) Shows the original time series with a linear trend, and (b) is where the trend is removed to reveal the similarity between the time series. 19

Figure 2.9 Smoothing can produce time series with less irregular movements. (a) Shows the original time series and (b) is where the time series after smoothing. 19

Figure 2.10 Dendrogram for hierarchical clustering. 22

Figure 2.11 K-means clustering with $k = 2$. 24

Figure 2.12 DBSCAN with $\mathtt{MinPts} = 4$, dashed circles denote ϵ-neighborhoods, x_1 is a core object; x_2 is directly density-reachable from x_1; x_3 is density-reachable from x_1; x_3 and x_4 are density-connected; and x_5 is an outlier. 26

Figure 2.13 The global parameter for DBSCAN cannot identify varying densities within a cluster. 27

Figure 2.14 OPTICS example with $\mathtt{MinPts} = 4$, core-distance is ϵ', and reachability-distances marked with arrows. 28

Figure 2.15 (a) Three similar, but unaligned time series, (b) average time series with Euclidean sum divided by the number of sequences, (c) averaged time series with DBA method with the second (peaking) time series as the initial template. 31

Figure 2.16 Proton Speed (ICME Speed) plotted against Plasma-Beta of the 181 ICME events used in this dataset. 38

Figure 2.17 Illustration of the feature extraction phases steps. (a) Shows an illustration of an image taken on board of the SDO which shows the vector magnetic field that is further used to detect the active regions in (b) that are masked in (c) before computing the magnetic field related features. [1] 41

Figure 2.18 The slicing process for the SWAN-SF benchmark dataset. For a given active region, each observation window (blue bars) is labeled (on the left, in green) according to the intersection (black circles) of its prediction window (red bars) with the magnitude of the largest flare reported for this active region in that time window. Note that each observation window produces one MVTS. 44

Figure 3.1 A comparison between the standard DTW and segDTW, (a) is the mapping of standard DTW, which is computed linearly and is showing signs of a pathological warping path, (b) shows the mapping results of segDTW with peak identification, here each segment can be simultaneously computed. Hence, segDTW can generate faster and more intuitive results. 47

Figure 3.2 (a) Standard DTW mapping with three pairs of peaks identified. (b) Dissimilarity matrix used for segDTW with identified peaks, the warping path can only traverse through the colored dynamic sub-windows. 52

Figure 3.3 Two time series computed with segDTW with (a) peak, with distance 16.16 (b) valley, with distance 18.75, and (c) equal-window, with distance 22.14, respectively. 54

Figure 3.4 The relationship between the number of peaks with accuracy for 16 datasets. 56

Figure 3.5 Datasets are in ascending order of time series length, (a) shows the processing time for standard DTW, DDTW, and segDTW on the logarithmic scale, with segDTW speedup becoming more apparent as time series length increases; (b) shows the corresponding accuracy performance. 58
Figure 3.6 Warping path of (a) standard DTW, (b) segDTW, and (c) segDTW combined with the Itakura warping window 59
Figure 4.1 Similarity plot of (a) nearest time series seed and (b) furthest time series seed. Visually, the strongest bend is the steepest slope, which occurs between time series 'd' and 'c' in (a), and between time series 'd' and 'b' in (b). This bend is conceptually a virtually sparse region, and a suitable space to partition the dataset/cluster. . . . 63
Figure 4.2 The Accuracy, F-score, and Rand Index of 50 datasets from the UCR repository. Using Distance Density clustering, the orange dot is the performance of furthest seed initialization; the blue dot is the performance of nearest seed initialization, and the box region is the performance range of k-means with random initialization. . 67
Figure 5.1 Triangle inequality theorem: given a proper triangle with side lengths a, b, and c that are all positive and excludes the degenerate case of zero area, the sum of any two sides is greater than the remaining side. 68
Figure 5.2 A comparison between minimal DDC and class label assignment, here minimal clustering refers to producing the same number of clusters as there are class labels, (a) shows the Silhouette Coefficient, (b) shows the Dunn Index, and (c) shows the Davies-Bouldin Index. For all three indices, minimal clustering showed better performance than the ground truth class label assignment. . . . 70

Figure 5.3 Three types of distribution, (a) multiple classes belonging to the same natural cluster, (b) class label coincide with natural clusters, (c) multiple natural clusters forming the same class. 70
Figure 5.4 Evaluation of clustering with internal indices values and the corresponding accuracy. 72
Figure 5.5 Accuracy and descriptive power in relation to the number of clusters. 73
Figure 5.6 Evaluation of clustering with variance values. 75
Figure 6.1 Dendrogram of four implementations of AGNES, and three implementations of OPTICS. 78
Figure 6.2 Cophenetic coefficient matrix showing the similarity between dendrograms from different hierarchical clustering methods for dataset "Coffee". 80
Figure 6.3 Goodman and Kruskal's gamma matrix showing the similarity between dendrograms from different hierarchical clustering methods for dataset "Coffee". 80
Figure 6.4 The training and testing accuracy curves for datasets: "BirdChicken", "Coffee", and "Meat", for the different number of clusters. 82
Figure 6.5 Distance Density accuracy curve for 16 parameters with AUC labeled. 86
Figure 6.6 Shape-based multivariate time series decision tree for DDC. . . . 87
Figure 6.7 AGNES accuracy curve for 16 parameters with AUC labeled. . . 89
Figure 6.8 Shape-based multivariate time series decision tree for AGNES. . 90
Figure 6.9 10-fold accuracy performance comparison boxplot for discrete data, AGNES, and DDC. 91

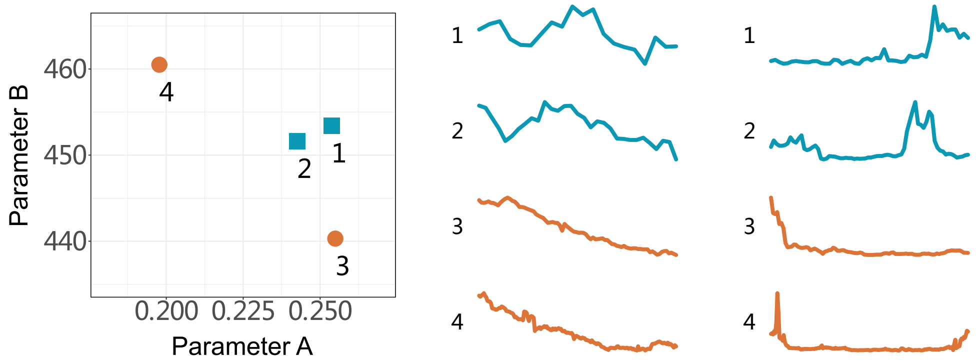
Figure 6.10 Accuracy curve for 16 parameters on both training and testing datasets of Prior 12 Span 06, with AUC labeled, and top five AUCs highlighted in red boxes. 93
Figure 6.11 Shape-based multivariate decision tree for Prior 12 Span 06 solar flare dataset. 94
Figure 6.12 Boxplots of cluster accuracy for (a) original data, (b) offset translation, (c) amplitude scaling, (d) detrend with difference and (e) detrend with log. 97
Figure 6.13 The original 300 pre-flare time series of (a) 100 C-class, (b) 100 M-class, and (c) 100 X-class of parameter TOTUSJZ. 98
Figure 6.14 5 clusters of parameter TOTUSJZ with offset translation. 99
Figure 6.15 Decomposed cluster from one of five clusters from parameter TOTUSJZ. In this particular cluster, there are a total of 18 pre-flare time series, 1 C-class, 5 M-class, and 12 X-class. The normalized time series and their respective averages are in the top row, (a) the entire time series cluster, (b) C-class pre-flare, (c) M-class pre-flares, (d) X-class pre-flares. The time series in this particular cluster but in the unnormalized form is shown on the bottom row, (e) the entire cluster of unnormalized time series, (f) C-class pre-flare, (g) M-class pre-flare, (h) X-class pre-flare. 100

1 INTRODUCTION

Supervised learning and unsupervised learning are the two main types of tasks in the field of machine learning. Supervised learning is usually performed in the context of classification, where prior knowledge is used to obtain a classification standard that approximates the relationship between the input and the output data. Unsupervised learning, on the other hand, does not need labels to process and is used to discover natural structures within a dataset. A common method for unsupervised learning is clustering; it is an important part of exploratory data mining and can be regarded as a form of learning activity, where certain features of data are learned and generalized.

Clustering analysis is a well-studied topic in the data mining community [2]. Essentially, clustering is the task of grouping objects together where high intra-cluster similarity and low inter-cluster similarity are achieved. Clustering originated in anthropology [3], and was later introduced to psychology [4, 5]. Now clustering is widely used in biology, medicine, business, criminology, and many other fields where the natural distribution of data can help us gain insight and obtain knowledge. In addition to being a stand-alone exploratory method, clustering can also be applied as a pre-processing step or subroutine to many data mining tasks, such as rule discovery, indexing, summarization, anomaly detection, and classification [6].

With the advancement of data collection and storage, time series data is now widely acquired in many disciplines. Typically, time series data is a succession of measurements equally spaced in time that can describe the course of events, which leads to shape-based comparisons. Fig. 1.1 (a) is the distribution of summarized averaged time series in discrete form, described using parameters A and B. Here, events 1 and 2 have the same label, and events 3 and 4 have the same label. Fig. 1.1 (b) and (c) shows the time series sequence of parameters A and B, respectively. While the data distribution is more brief and direct in Fig. 1.1 (a), the natural grouping or separation based on the summarized values in the Euclidean space, may not convey the true event labels. In contrast, the time series representation, shown in Fig. 1.1 (b), and (c) are ample in detail, and as a result, the separation between groups of event data is much more distinct.



(a) Averaged distribution. (b) Parameter A. (c) Parameter B.

Figure 1.1: (a) The discrete averaged time series distributed in the Euclidean space, (b) original time series for parameter A, and (c) original time series for parameter B.

This thesis presents a pipeline for time series clustering. We will focus on the similarity measurement, clustering heuristic, and cluster quality assessment for the specific and increasingly widely applied data type, time series. As the groundwork for time series comparison, we present our contribution of utilizing segmentation as a means of improving the traditional Dynamic Time Warping algorithm. Being the core of unsupervised learning, we include the exploration of the existing clustering and evaluation techniques in addition to introducing our novel time series clustering approach and its subsequent evaluation.

1.1 Motivation

Essentially, time series are sequences of measurements and are therefore innately high dimensional, which presents both challenges and opportunities with unsupervised learning. On the one hand, time series data can describe an event in much more detail than a discrete summarized statistical figure, such as the mean or median. Conversely, much of the previous work on machine learning was not carried out with time series data in mind. In practice, the time and space complexity for time series processing is always a bottle-neck for efficient problem solving.

The emerging usage of time series data injects additional interest in machine learning. Clustering is a prevalent data exploratory branch of machine learning; its fascination lies in ubiquitous applications and endless possibilities. Based on the data and goal, the clustering procedure could be broad or specific. Clustering is effective in extracting information and knowledge from data, which is one of the primary goals of data mining. Over the years, various clustering algorithms have been proposed, targeting different data types and different data spaces. Although the clustering of discrete datasets can be considered a fairly solved problem, with numerous well-established methodologies, existing clustering approaches often do not perform well when applied to time series data, especially in the multidimensional case [7]. With the emerging applications of time series data, the study of time series clustering and its supporting mechanisms are imperative.

1.2 Challenges

A clustering process consists of three main steps: the similarity measure, the clustering methodology, and the quality assessment; all of which presents its own unique predicaments. When dealing with time series clustering, there are usually two routes. The first is trying to lower the dimensionality of the data, the merit of this method is the efficiency, while the elimination of too many details negates the incentive to use time series data. The other direction of time series clustering is to use more adaptive similarity measures, which could maximally retain data details. However, due to the high dimensionality, this type of method is typically time-consuming. Furthermore, the similarity of time series is not a true distance and does not satisfy the triangle inequality theorem. Which means the clustering heuristic and evaluation methods of time series cluster is different from that of discrete data.

Depending on the data type and amount, clustering can range from naive to rigorous. Simple datasets such as the famous Iris dataset is 4.4 kilobyte [8], and takes very little time to process. While one day of ECG (electrocardiogram) data of one patient is over 1 gigabyte [9], and the SDO (Solar Dynamics Observatory) returns over 1 terabyte per day [10]. With improved storage and processing power, the accumulation and usage of time series data have become a possibility, and there is no shortage of valuable data sources. Due to the size and details of time series datasets, effective and efficient analysis has become the core of the issue. Since most real-world application datasets are not labeled, clustering would be a useful approach to convert the collected data into knowledge.

What adds to the adversity is that the notion of clustering is not a single specific algorithm, but rather a general task of minimizing intra-cluster differences and maximizing inter-cluster differences. The possible combinations of cluster configuration are computationally prohibitive [11, 12], meaning most clustering algorithms are user and data-dependent. Given the imprecise definition, there is no shortage of cluster heuristics or the evaluation of said heuristics.

1.3 Outline

The rest of this thesis is organized as follows. In Chapter 2, we introduce the background on elastic similarity measures, cluster heuristics, cluster representation, cluster evaluation, and the associated datasets. This serves as the basis of further discussion of our work regarding time series clustering. In Chapter 3, we present our intuitive, scalable, and highly efficient elastic similarity measure: segmented-Dynamic Time Warping (segDTW). Based on significant features, we are able to add a layer of global similarity identification before the more detailed mapping. In addition to generating more intuitive results, segDTW also improves the efficiency of similarity comparisons. Then in Chapter 4, we explain Distance Density Clustering (DDC), a deterministic clustering method we developed for time series data. The idea is to identify virtual sparse regions within a time series dataset, and further re-balance clusters with similarity information. Chapter 5 discuss how variance can be used to evaluate time series clusters.

In Chapter 6, we first discuss the correlation and distinction between different hierarchical clustering methods. Then presents the usage of decision trees to aggregate shape-based univariate clustering results, which can be used to make multivariate decisions. Furthermore, we discuss how we used normalization to emphasize shape similarity and how cluster profiles can be used in real-time classification. Finally, in Chapter 7, we conclude this thesis with a summary of our findings and discuss some future directions of our work.

2 BACKGROUND

In this chapter, we introduce the necessary background information associated with time series clustering. When analyzing time series data, one of the essential issues is the similarity measure. In Section 2.1, we introduce the basics of Dynamic Time Warping [13], a widely used elastic measure that has shown to be effective for sequence data comparisons. In addition to the original DTW algorithm, we also discuss several DTW improvement methods as well as time series normalization techniques. Then in Section 2.2, we discuss some commonly used clustering heuristics, which will serve as a precursor to the later introduction of our shape-based clustering algorithm. Another obstacle for time series clustering is the representation of time series clusters. In Section 2.3, we review the Dynamic Time Warping Barycenter Averaging [14] technique, which is a global averaging technique specifically geared toward time series data. The final step of cluster analysis is its evaluation; Section 2.4 briefly discusses some commonly used internal and external indices, as well as methods to compare dendrograms for hierarchical clustering. Lastly, in Section 2.5, we review the datasets that are used in the experiments supporting our work.

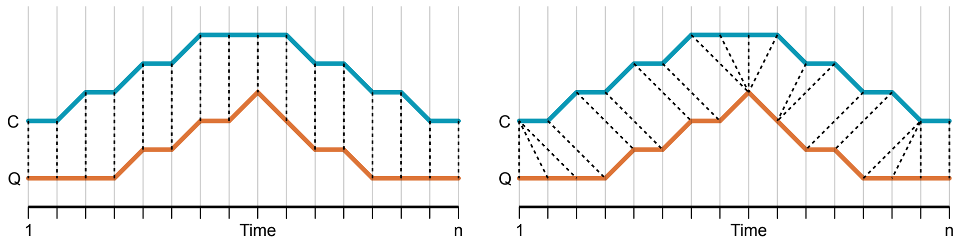
2.1 Similarity Measure

Time series data is widely applied in a variety of domains, such as voice recognition, the stock market, solar activities, medical research, and many other scientific and engineering fields where analysis of real-valued, continuous data are more important than the generalization of an event. Time series data enjoys its popularity because it records details that are commonly overlooked by the summarization of data. Since similarity measure is specific to data type and applications, a suitable similarity measure needs to be applied for time series data.

Generally, similarity measures can be categorized as either lock-step or elastic. Lock-step measure refers to the Minkowsky distance, or L_p norm, meaning that the *i*-th element from one sequence is always mapped to the *i*-th element in the compared sequence. Realistically, real-world data are rarely pristine, meaning even if two sequences describing the same class of events are of the same duration, misalignment due to the temporal discrepancies can be very common. When the mapping between two sequences is fixed, lock-step measures are sensitive to noise and misalignments [15]. Thus, while Euclidean distance is commonly applied in everyday life and various domains, it is generally inadequate for sequential data comparisons.

In contrast to the lock-step measure, elastic measures allow one-to-many or even one-to-none mappings [16]. A popular elastic measure algorithm is the Dynamic Time Warping (DTW). DTW can evaluate the similarity between two temporal sequences, which may vary in time or speed and has been widely accepted as an efficient measure for time series data [17–20]. Generally, this is a method that can allow computers to find an optimal match between two given sequences under certain restrictions. Its advantage is that it allows flexible one-to-many mappings; in other words, DTW allows one point from the query sequence to be mapped to multiple points in a candidate sequence or visa versa. Originally, DTW was used in speech recognition; later, it was adapted to various real-world data mining problems. Fig. 2.1 shows the mapping of Euclidean distance and the Dynamic Time Warping measure.

Equations 2.1 and 2.2 are the Euclidean and Dynamic Time Warping [13] distances respectively, where given two time series sequences Q and C, with $Q = \{q_1, q_2, ..., q_i, ..., q_n\}$ and $C = \{c_1, c_2, ..., c_j, ..., c_m\}$. When calculating the Euclidean distance, the total distance is



(a) Euclidean Distance (b) Dynamic Time Warping

Figure 2.1: The mappings of lock-step and elastic similarity measures: (a) Euclidean distance and (b) Dynamic Time Warping.

the sum of the distance between each of the corresponding one-to-one mappings between q_i and c_i. In the case of DTW distance, a n-by-m distance matrix is first constructed containing the distance information between all the elements from both query and candidate sequences. The warping path is denoted as $W = \{w_1, w_2, ..., w_k, ..., w_K\}$, and while there are exponentially many warping paths, only the path minimizing *Dist(DTW)* is of interest [21]. DTW works by mimicking shape matching, its effectiveness in finding similar shapes in data is due to the algorithm's ability to look within a certain range for a local optimum mapping. Various step patterns are used as a means to introduce weight to the warping path. Eq. 2.3 can be considered a standard step pattern.

$$\mathrm{Dist}(\mathrm{Euclidean}) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \tag{2.1}$$

$$\mathrm{Dist}(\mathrm{DTW}) = \min\{w_k(Q, C)\} \tag{2.2}$$

$$D(A_i, B_j) = \delta(a_i, b_j) \qquad + \min \begin{Bmatrix} D(A_{i-1}, B_{j-1}) \\ D(A_i, B_{j-1}) \\ D(A_{i-1}, B_j) \end{Bmatrix} \tag{2.3}$$

Another way to visualize the effect of DTW is shown in Fig. 2.2 using the same time series from Fig. 2.1. The DTW warping path is shown in a bold black line, which shows how a warping path traverse through the similarity matrix and demonstrates how one-to-many mapping works. In comparison, the diagonal dotted gray line is the one-to-one mapping that corresponds to the Euclidean distance.

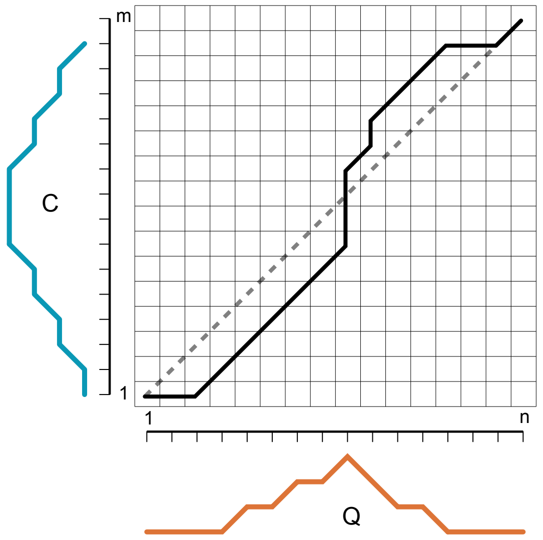


Figure 2.2: The Dynamic Time Warping warping path between time series Q and C.

In order for DTW to effectively map and calculate the similarity between any given sequences, certain restrictions need to be met. Fig. 2.3 shows the violation of the three basic conditions for the DTW algorithm: the *boundary condition*, *monotonicity*, and *continuity* [22]. The boundary condition means that the first and last components from two compared sequences are always mapped, and because DTW only allows one-to-many mapping and not one-to-none mappings, every element has to have a mapping component. Mapping within a pair of time series can not cross, as it would not make sense on the time scale.

Monotonicity means time can only proceed in one direction and that the warping path cannot go back in time. The continuity constraint is also known as the step pattern constraint; it is where the warping path can only follow the predetermined steps patterns and cannot make any jumps. The continuity constraint is also a form of eliminating one-to-none mapping and implies monotonicity.

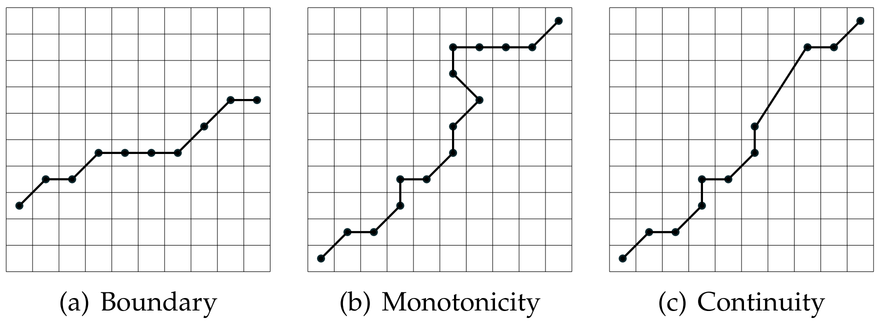


Figure 2.3: Violation of the three basic conditions for DTW algorithm: (a) the boundary condition, (b) the monotonicity condition and (c) the continuity condition

The major bottleneck of DTW computation is the time and space complexity. Calculation of the cost matrix requires $n \cdot m$ operations that can be done in constant time, hence $O(N^2)$, where $N = n = m$. With many datasets, the sequences of interest could be relatively lengthy, meaning the quadratic time complexity is nontrivial. In addition, since DTW requires the entire cost matrix to construct the optimal warping path, the space complexity of the algorithm is also quadratic, $O(N^2)$. This high complexity is particularly important since time series are high-dimensional data and the polynomial space complexity would quickly exhaust gigabytes and even terabytes of memory [23].

Before DTW in more details, we first propose three key elements of time series similarity, which are *range value similarity*, *duration similarity*, and *shape similarity*. Individually, none of these three elements can guarantee similarity for time series. Time series need to achieve high similarity in all these three aspects to be considered truly similar. In particular:

- Range value similarity refers to the absolute value range of time series. Range value similarity can be considered as vertical similarity; however, it does not promise actual similarity, but rather a comparable range. This can serve as a rough categorization, or threshold to differentiate large amount of data.

- Duration similarity can be considered a horizontal range value, referring to the time series measurement duration. Two events may have similar range values and shapes; however, if one lasted only seconds while the other lasted decades, then in most circumstances, they are not considered similar, unless a specific task calls for such usage.

- Shape similarity refers to the contour or shape of the time series. The shape similarity is an important part of the similarity elements; while used alone, it cannot guarantee similarity, there is most likely no similarity between time series when shape similarity is not present. In practice, shape similarity can be paired with range value similarity or duration similarity or both.

When all the elements of similarities mentioned above are high, we can conclude that the examined time series are highly similar. This is not to say that they have to occur simultaneously to be meaningful: each aspect of similarity could be individually significant under certain contexts. For example, when credit card companies analyze customer spending, both a big spender and an average spender would likely have higher spending in the holiday season, which could generate similar spending trends. While the shape can be useful when identifying seasonal spending trends, if the credit card company is interested in the customers' spending power, then the range value is of higher priority than the shape of the spending trend.

2.1.1 DTW Improvement Techniques

Although widely applied with time series data today, DTW is by no means a new similarity measure. It was originally applied to the obvious sequential data of speech recognition decades ago. By allowing elastic mapping, DTW provides more intuitive alignment between sequential data such as time series. Despite its general success, DTW often attempts to explain the variability on the y-axis of the similarity matrix by warping on the x-axis. This compensation is the cause of the undesirable phenomenon of singularity and could lead to pathological warpings [24]. When pathological warpings occur, the DTW algorithm can no longer provide us with sensible or reliable results.

For DTW algorithm improvement, both quality-wise and efficiency-wise, many methods have been proposed. Overall, time series data analysis can be categorized as complete sequence analysis or sub-sequence analysis. Complete sequence analysis improvement techniques can be further categorized as global constraints and approximations. The goal of global constraints is the attempt to place restraints of global influence on the warping path, which could eliminate computations and therefore shorten the processing time. Time series data are high-dimensional data, and approximations are the attempt to manipulate the input time series as a means to lower data dimensionality and cut-down on the number of comparison computations. In the extreme case, one time series sequence can be abstracted to one number, such as the mean or median of the sequence; this type of statistical summarization is commonly used in traditional non-time series data analysis. Here we discuss some of the more commonly applied DTW improvement methods for complete sequence analysis.

Windowing is a global constraint, and has been used by different researchers before it was formally summarized by Berndt and Clifford [25]. Windowing effectively prunes the corners of the distance matrix, where the warping path is not allowed to venture to, meaning the potential warping path is restrained to a fixed region. This method

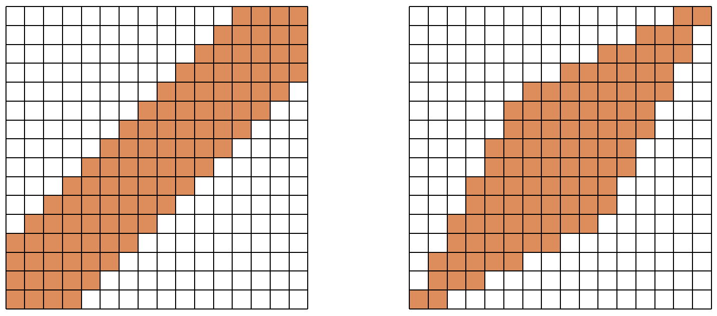

(a) Sakoe-Chiba Band (b) Itakura Parallelogram

Figure 2.4: DTW windows: (a) Sakoe-Chiba Band and (b) Itakura Parallelogram, only the colored cells will be computed, thus improving efficiency. Since the warping path is constrained, pathological warpings can be prevented to a certain degree.

can alleviate some of the singularity problems, but cannot prevent it [26]. Two of the well-known global windowing constraints are shown in Fig. 2.4, which are the *Sakoe-Chiba Band* [27], which is a slanted diagonal window, and *Itakura parallelogram* [28], which is a parallelogram-shaped window. The window sizes can be adjusted according to user requirements. When the windows are minimum in size where only the straight diagonal path is allowed, and DTW becomes Euclidean distance.

Slope weighting encourages the warping path to progress in a certain direction, typically to remain close to the diagonal. Shown in Eq. 2.4, when a weighting factor W is generally applied to the whole sequence, it is a global constraint. Depending on the specific weighting factor, it reduces the frequency of singularities [29]. The weight factor would need to be specified *a priori*, depending on user specifications.

$$D(A_i, B_j) = \delta(a_i, b_j) \qquad + \min \begin{Bmatrix} D(A_{i-1}, B_{j-1}) \\ WD(A_i, B_{j-1}) \\ D(A_{i-1}, B_j) \end{Bmatrix} \tag{2.4}$$

Step pattern is a global constraint approach that encourages changes to the warping path to avoid pathological warpings. Fig. 2.5 shows the four commonly used step patterns.

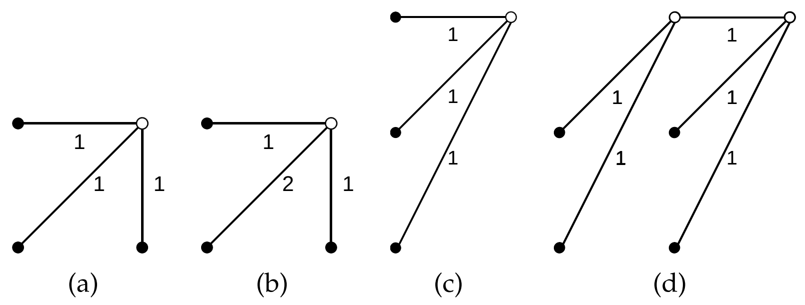


Figure 2.5: Step Patterns: (a) *symmetric1* is the basic step pattern, (b) *symmetric2* favors the diagonal warping path, (c) *asymmetric* limits time expansion to a factor of two, and (d) *rabinerJuangStepPattern* has attributes local continuity constraint type, slope weighting, and the state of smoothed or not.

Based on symmetry and slope bounds, Sakoe and Chiba proposed *symmetric1*, *symmetric2* [27], and *asymmetric* [30] approaches. The first is a basic step pattern. The second favors the diagonal warping path similar to slope weighting, and the third limits time expansion to a factor of two. Rabiner and Juang introduced *rabinerJuangStepPattern* [31] based on the continuity constraint, slope weighting, and the state of being smooth or Boolean. We will be using the *symmetric1* step pattern for segDTW and all the experiments supporting this work.

Lower bounding is another global constraint for improving the DTW algorithm. By defining tight and fast lower bounding functions, sequences that cannot provide a better match in the process of finding the warping path are pruned. The idea is to favor the execution time needed for calculating the similarity matrices on large datasets. While Yi et al. [32] gave an approximation for indexing, the lower bounding function introduced in LB_Kim [33] was the first to define an exact indexing. Compared to the earlier works, LB_Keogh [19] had an overall greater pruning power and could also give tighter bounding measures. Their lower bounding function is defined based on U and L, the two new time series generated from the reference time series Q, such that $U_i = max(q_{i-r}, q_{i+r})$ and $L_i = min(q_{i-r}, q_{i+r})$. Where $j - r \leqslant i \leqslant j + r$, and r is used to define the allowed warping range. Having the bounding envelope defined by U and L, the lower bounding function is defined as "the squared sum of the distances from every part of the candidate sequence C not falling within the bounding envelope, to the nearest orthogonal edge of the bounding envelope" [19].

Among other variants of the DTW method, PDTW (piece-wise DTW) [34], DDTW (derivative DTW) [24], and shapeDTW [35] are some of the more popular methods which attempt to manipulate the input time series to improve either the warping path quality or the processing time. The primary achievement of PDTW is to increase the speed factor by one to two orders of magnitude on average, while maintaining the accuracy of DTW, by utilizing a piece-wise approximated representation of the time series instead of the raw data. Similarly, DDTW utilizes an approximated derivative of the time series to work on a higher level of similarity between two time series. A similar approach is also used in shapeDTW. Zhao et al. [35], where each temporal point q_i of a time series Q is represented by a shape descriptor that encodes the structural information of a fixed-width neighborhood of q_i. The choice of the descriptor depends on the general structure of the time series and the user requirements. Some of the widely used descriptors are the slope, piece-wise aggregate approximation (PAA), discrete wavelet transform (DWT), and the histogram of oriented gradient for 1-D time series (HOG1D).

The methods discussed so far are based on either global constraints, data abstraction, or indexing [36]. Windowing, slope weighting, and step pattern approaches fall into the constraint category, whereas heuristics such as DDTW, shapeDTW, and pDTW, belong to the data abstraction category. The idea behind the constraints category is to avoid pathological warpings by trying to encourage the warping path to stay close to the diagonal rather than to stray excessively vertical or horizontal. The abstraction family of methods uses coarsely granulated data to avoid detailed pitfalls that could potentially cause pathological warpings. The third category that revolves around lower bounding is also referred to as indexing. The methods which utilize indexing are different from the other two in the sense that they are mainly focused on speeding up the algorithm. It is important to note that constraints and data abstraction improve the time and space complexity only by a constant factor since they still have to deal with the cost matrix, which results in both the time and space complexity remaining quadratic. However, the third category made a significant contribution in this aspect by bringing the computation time and space down to $O(N)$ using different indexing methods [19, 33].

2.1.2 Normalization

In the context of data mining, normalization refers to the scaling of data attributes so that the data are restricted to a smaller vertical range. Normalization is generally required when we work on attributes with different scales. For example, when the magnitude differences are significant, the progression of events, or the shape of measurements, can often be overlooked. In actuality, the shape of certain past behaviors could be a good indication of certain future behaviors or events. By using normalization on time series data, we could focus on the shapes of physical parameters without the interference of value differences.

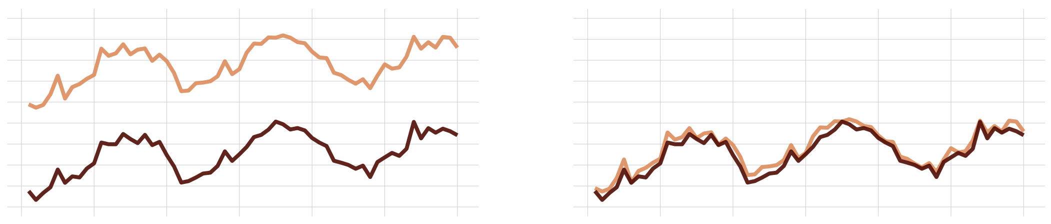
In addition to the scale adjustment, time series normalization also refers to the shifting and scaling of data to eliminate the effect of gross value influences. The four most commonly applied normalization techniques for time series data are *Offset Translation*, *Amplitude Scaling*, *Trend Removal*, and *Smoothing* [37].

Offset Translation

Offset translation means the shifting of time series. Offset is a signal processing term, used when sequences are similar in shape but are within different ranges. Shown in Eq. 2.5, offset translation means subtracting the mean from the original time series, namely,

$$ts = ts - mean(ts) \tag{2.5}$$

Here the mean value is computed for each time series individually and is simply the average over all the values in that specific time series. The translation of the offset can be useful for similarity comparisons. However, an immediate drawback of this operation is that the range values would be overlooked since the value differences are removed. For stored (not real-time) time series, the offset can be removed by subtracting the mean amplitude from each sample. An example of an offset translation operation on a time series is shown in Fig. 2.6.



(a) Before offset translation

(b) After offset translation

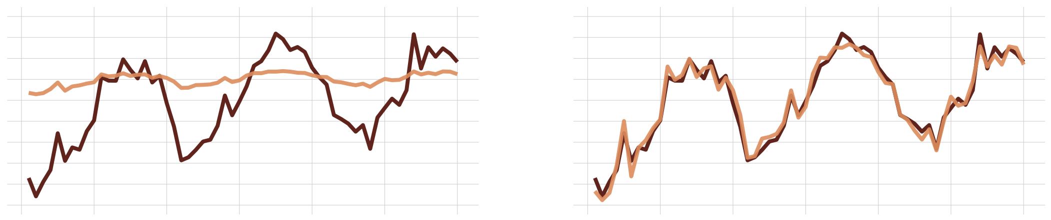
Figure 2.6: Offset translation shifts the range of time series to focus on the shape similarities. (a) Shows the original time series, (b) shows the normalized result after offset translation is applied.

Amplitude Scaling

Amplitude is also a term borrowed from signal processing. It measures how far, and in which direction, does a variable differ from a defined baseline. Scaling of a signal's amplitude means changing the strength of the signal. With time series data, we remove the different amplitudes in hopes of finding similarity by excluding the physical parameter's strength.

$$ts = (ts - mean(ts))/std(ts) \tag{2.6}$$

Shown in Eq. 2.6 and illustrated in Fig. 2.7, amplitude scaling is achieved by first moving time series by its mean and normalizing the amplitude by the standard deviation. Which means that in a way, offset translation is included in amplitude scaling. In fact, when $\mathtt{std}(\mathtt{ts}) = 1$, the two methods are identical.


(a) Before amplitude scaling

(b) After amplitude scaling

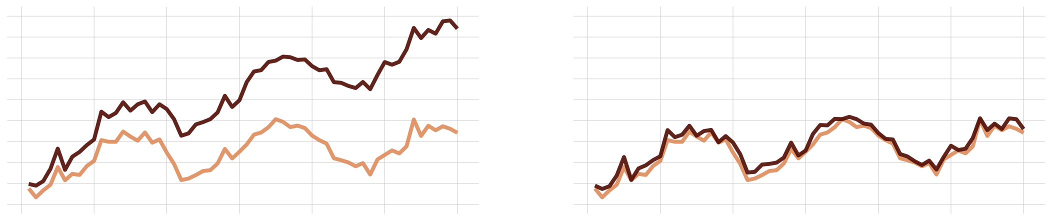
Figure 2.7: Amplitude scaling changes the strength of the signal (time series) to make shape similarities easier to identify. (a) Shows the original time series, (b) shows the time series after amplitude removal.

Trend Removal

Trend removal is regularly used in prediction. Trends represent long-term movements in sequences. In identifying patterns in sequence data, trends may become distracting, and therefore, it is often justified to remove them for revealing possible oscillations. To this end, the regression line of the time series needs to be identified and then subtracted from the time series. An example of linear trend removal is shown in Fig. 2.8. However, unlike offset translation and amplitude scaling, trend removal is not a straightforward operation. There can be different types of trends or even multiple trends. In our experiments, we only consider linear and logarithmic trends.

Smoothing

Smoothing is usually performed with a moving window on the time series to obtain the average values of each data point with those of its neighbors. Shown in Fig. 2.9, smoothing can eliminate some irregular movements, but is sensitive to outliers and also invalidates data at the beginning and end of the time series.

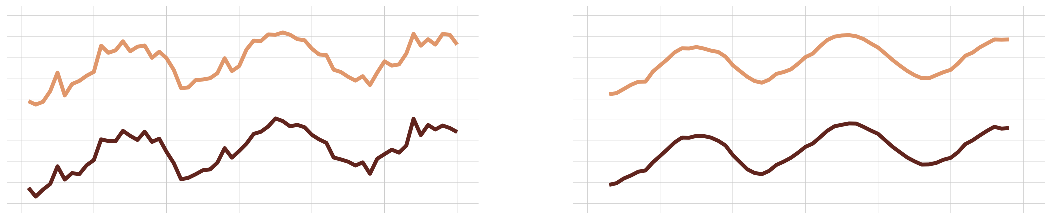


(a) Before trend removal

(b) After trend removal

Figure 2.8: Trend removal removes the patterns, or trends, from time series data in the hope of focusing on the value changes without the long-term trend distraction. (a) Shows the original time series with a linear trend, and (b) is where the trend is removed to reveal the similarity between the time series.

When time series are relatively short in length, smoothing is typically unsuitable. On the account that an effective smoothing window could shorten the time series excessively, rendering the result ineffective.



(a) Before smoothing

(b) After smoothing

Figure 2.9: Smoothing can produce time series with less irregular movements. (a) Shows the original time series and (b) is where the time series after smoothing.

2.2 Clustering

Clustering analysis is an exploratory data mining task and therefore does not refer to a specific algorithm. The appropriate choice for a clustering algorithm and its corresponding parameter settings would depend on the data and intended application. Since there are multiple notions of clustering, there are various ways to classify clustering algorithms, some of which overlaps.

First and foremost, based on separation, clustering can be categorized as crisp clustering or fuzzy clustering. Crisp clustering is also referred to as exclusive clustering, meaning an observation either belongs to a cluster or not. Fuzzy clustering is also referred to as non-exclusive clustering, in which case each observation can belong to a cluster to a certain degree. Additionally, clustering can be roughly categorized as hierarchical, partitioning, density-based, grid-based, model-based, and multi-step clustering algorithms [38].

Based on the partitioning criteria, clustering can be categorized as single level clustering or hierarchical clustering. Single level clustering refers to noninclusive cluster results, whereas hierarchical clustering produces nested clusters. Examples of hierarchical clustering includes Hierarchical Agglomerative Clustering [39], Hierarchical Divisive Clustering [39], and OPTICS [40]. Hierarchical clustering is known for its strong visualization capabilities, as well as not requiring the number of clusters to be preset. Additionally, it is straightforward to adapt different similarity measures as well as different data types to hierarchical clustering.

Based on similarity measure, clustering can be categorized as partitioning or density-based. One of the most widely used partitioning algorithms is k-means [41], which can also be categorized as distance-based clustering. K-means uses a cluster center to represent a cluster and operates with the goal of minimizing intra-cluster distances. Depending on the representation of cluster center and cluster center selection, k-means have variations such as k-medoid [42] and k-means++ [43]. A good representation of a density-based model is DBSCAN [44] and OPTICS [40], which are also known as connectivity-based.

Grid-based clustering is related to density-based clustering, and it works by partitioning data space into a finite number of cells that form a grid structure. Then clusters are formed based on the regions where data points are denser than its surroundings [45].

However, with grid-based clustering, the cell sizes, borders, and density threshold are all predefined. Furthermore, because the grid sizes are uniform, it can be difficult and inefficient when handling irregular data distributions. The idea of grids were originally proposed by Warnekar and Krishna [46], some popular grid-based clustering methods include STING [47], CLIQUE [48], and WaveCluster [49].

Unlike hierarchical and k-means, where the distribution of data is not based on existing models. Model-based clustering assumes the distribution is a mixture of two or more clusters, and attempt to optimize the fit between data and some probabilistic model [45]. Model-based clustering uses crisp/fuzzy cluster assignment [50].

With the goal of improving clustering performance, attempts of combining existing models to form multi-step clustering models were made. Multi-step models either incorporate different clustering algorithms during different stages of clustering or use multiple data resolution for a tiered clustering structure.

Different clustering approaches are suited for different types of data, and for any clustering algorithm, there are always counter-examples where the method does not work. Clusters are, in large part, in the eye of the beholder [51], and one person's noise could be another person's signal [52]. This also complies with the No Free Lunch theorem [53]. Here we will briefly discuss some of the most widely applied cluster heuristics.

2.2.1 Hierarchical Clustering

Hierarchical clustering tries to separate data into different levels that have a top to bottom ordering and builds corresponding tree structures called dendrograms. Shown in Fig. 2.10, there are two types of hierarchical clustering, agglomerative, also known as Agglomerative Nesting (AGNES), and divisive, also known as Divisive Analysis (DIANA). AGNES is a bottom-up approach, where each event is assigned as its own cluster, and are aggregated into larger clusters based on the similarity between each cluster. This process is repeated until all events form one single cluster containing the entire dataset. DIANA is a top-down approach, where all events originate as one cluster and are then partitioned to form two least similar clusters. This process is repeated until each event forms its own cluster. Due to its simplicity, AGNES is more commonly applied.

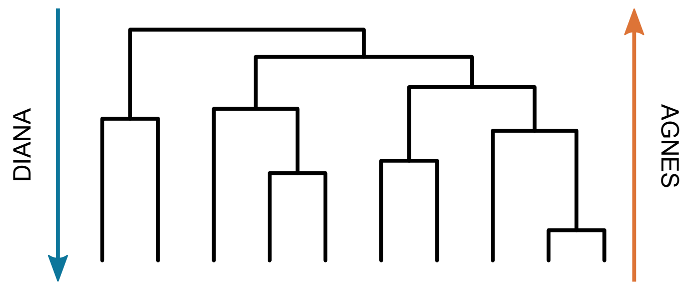


Figure 2.10: Dendrogram for hierarchical clustering.

The similarity measurement in both AGNES and DIANA is traditionally depicted using distance. Whereas OPTICS (ordering points to identify the clustering structure) [54] is a density-based hierarchical clustering method. OPTICS is an extension of DBSCAN (density-based spatial clustering of applications with noise), and it has a structure similar to AGNES. A major advantage of the hierarchical structure is its intuitiveness and great visualization power.

In an agglomerative structure, clusters are joint based on the similarity between elements or clusters. When comparing the similarity of clusters, various measures can be adopted. The cluster merging method in AGNES is called linkage. The most commonly used linkage measures are nearest, furthest, and average distance. Which corresponds to single link, complete link, and two types of average links, which are Unweighted Pair Group Method with Arithmetic mean (UPGMA) and Weighted Pair Group Method using Arithmetic mean (WPGMA). The results from hierarchical clusterings are not partitions, but rather a set of nested clusters that can be easily visualized as a dendrogram, which shows the merge of clusters at each step.

- Single link: shown in Eq. 2.7, the distance between two clusters is defined as the distance between the two nearest elements in each cluster. Clusters are usually longer and thinner, creating more chain-like clusters. Which means clusters of more arbitrary shapes could be identified.

$$d_{single}(A_i, B_j) = \min_{x \in A_i, z \in B_j} \mathrm{dist}(x, z) \tag{2.7}$$

- Complete link: shown in Eq. 2.8 the distance between two clusters is defined as the distance between two furthest elements in each cluster. Clusters are usually more spherical in shape, with relatively more compact borders.

$$d_{complete}(A_i, B_j) = \max_{x \in A_i, z \in B_j} \mathrm{dist}(x, z) \tag{2.8}$$

- Unweighted Pair Group Method with Arithmetic mean (UPGMA) [55]: shown in Eq. 2.9 the distance between two clusters is the average distance between all pairs of elements. Shown in Eq. 2.10, at each clustering step, the distance between a previously merged cluster $A \cup B$ with a new cluster C is the proportional averaging of the distance between A, C and B, C. More arbitrary shaped clusters can be identified, and clusters are usually more generally compact.

$$d_{avg}(A_i, B_j) = \frac{1}{|A_i||B_j|} \sum_{x \in A_i} \sum_{z \in B_j} \mathrm{dist}(x, z) \tag{2.9}$$

$$d_{(A \cup B),C} = \frac{|A| \cdot \mathrm{dist}(A, C) + |B| \cdot \mathrm{dist}(B, C)}{|A||B|} \tag{2.10}$$

- Weighted Pair Group Method using Arithmetic mean (WPGMA) [55]: is a modified version of UPGMA, the distance between clusters is the simple average. Shown in

Eq. 2.11 at each clustering step, the distance between a previously merged cluster $A \cup B$ with a new cluster C is simply the arithmetic mean of the distances between C and elements in $A \cup B$.

$$d_{(A \cup B),C} = \frac{\mathtt{dist}(A, C) + \mathtt{dist}(B, C)}{2} \tag{2.11}$$

2.2.2 K-means Clustering

K-means clustering was first introduced 50 years ago [2]. A simple example of the k-means algorithm is shown in Fig. 2.11. The aim is to minimize the intra-cluster sum of squares, by using the proximity of objects to the medoids of the clusters [56]. Drawbacks of k-means include its poor performance in the presence of outliers, and unsatisfactory results when clustering non-globular data. Based on the original k-means, k-medoid clustering was introduced by Kaufman and Rousseeuw [42], where instead of using the mean value, which could very likely be an artificial data point, it uses existing data points as clustering centers.

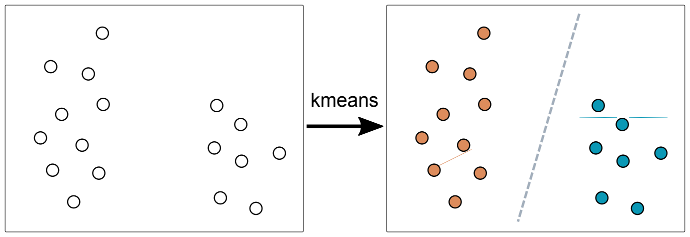


Figure 2.11: K-means clustering with $k = 2$.

As k-means is an NP-hard problem [57], finding the exact solution for large datasets is near impossible. The common approach is using Lloyd's algorithm, which finds the approximate solution. The largest issue with this approach is that it achieves a local optimum, meaning the seed initialization heavily influences the end result. With randomly chosen initial seeds, the quality of clustering could fluctuate. Therefore any meaningful help with initialization is better than random. Even though sometimes random initialization can result in higher accuracy, deterministic clustering can provide a stable algorithm with comparable results. This is especially important in applications, where stability and reproducibility are just as important as accuracy. To overcome the fluctuation of results due to random initialization, k-means++ was proposed [58], it is an approximation algorithm that can help to avoid some poor clustering for the original k-means. The idea behind the k-means++ algorithm is to initialize centroids (medoids) far away from each other, this speeds up the algorithm and provide better cluster results [58].

2.2.3 DBSCAN

Like the name suggests, DBSCAN (density-based spatial clustering of applications with noise) [44] is a well known density-based clustering method. A neighborhood is defined using parameters (ϵ, $MinPts$). Given dataset $D = x_1, x_2, ..., x_m$, we have the definitions:

- **ϵ-neighborhood**: for $x_j \in D$, its ϵ-neighborhood include points less than distance of ϵ, which is denoted as $N_\epsilon(x_j) = \{x_i \in D | dist(x_i, x_j) \leqslant \epsilon\}$.
- **core object**: x_j is a core object if the ϵ neighborhood of x_j has at least $MinPts$ points, denoted as $|N_\epsilon(x_j)| \geqslant MinPts$.
- **directly density-reachable**: if x_j is within the ϵ-neighborhood of x_i, and x_i is a core object, then x_j is directly density-reachable from x_i.
- **density-reachable**: for x_i and x_j, if exist sequences $p_1, p_2, ..., p_n$, of which $p_1 = x_i$, $p_n = x_j$, and p_{i+1} is directly density-reachable from p_i, then x_j is density-reachable from x_i.
- **density-connected**: if there exist x_k that is density-reachable for both x_i and x_j, then x_i and x_j are density-connected.

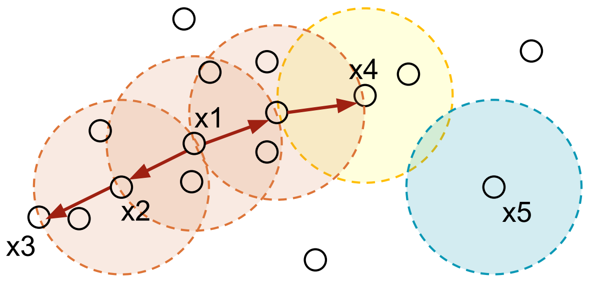


Figure 2.12: DBSCAN with $\mathrm{MinPts} = 4$, dashed circles denote ϵ-neighborhoods, x_1 is a core object; x_2 is directly density-reachable from x_1; x_3 is density-reachable from x_1; x_3 and x_4 are density-connected; and x_5 is an outlier.

Fig. 2.12 shows an example of DBSCAN, the cardinality of a neighborhood of radius ϵ has to exceed the minimum threshold number of points $\mathrm{MinPts} = 4$. x_1 is a core object, x_2 is directly density-reachable from x_1, x_3 is density-reachable from x_1, x_3 and x_4 are density-connected, and x_5 is an outlier.

In a DBSCAN cluster, objects are density-reachable from any core object of that cluster. Density connected objects are identified by iteratively collecting directly density-reachable objects. The ϵ-neighborhood is scanned for each object; if there exist more than MinPts objects, then a new cluster is formed. This process is repeated until all points have been checked, and no new objects can be added to this cluster.

2.2.4 OPTICS

Using a density network, DBSCAN is able to identify arbitrary shaped clusters; however, there is a drawback. Shown in Fig, 2.13, clusters *A*, *B*, and *C* can be easily found, but there are three visually distinct clusters *A1*, *A2*, and *A3* that are overlooked. DBSCAN cannot identify clusters of varying densities because there does not exist a global parameter setting for DBSCAN to accurately identify the cluster structures [40], which means that DBSCAN can only produce a single layered clustering result. In addition, DBSCAN shares a common issue with many clustering algorithms, which is that the parameters involved with clustering are determine *a priori*.

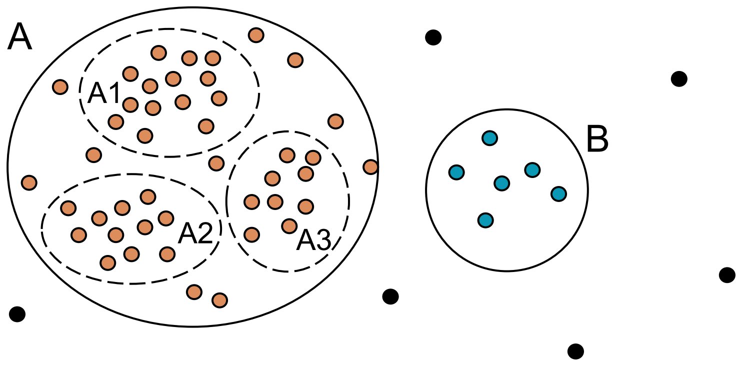


Figure 2.13: The global parameter for DBSCAN cannot identify varying densities within a cluster.

OPTICS is an extension of the DBSCAN; it mainly deals with varying densities, of which DBSCAN could not identify. OPTICS is where an infinite number of density parameters are processed at the same time. The events in a dataset are ordered so that the most spatially similar events become neighbors in order. OPTICS can be represented with a reachability-plot, which is a special type of dendrogram, and it can also be visualized as a traditional dendrogram.

Similar to DBSCAN, OPTICS also has two parameters (ϵ, $MinPts$). The difference is that the parameters are only used to improve the performance of the clustering process if the user has certain insight into the data beforehand, but they are not required.

Unlike DBSCAN, OPTICS does not assign cluster memberships. Instead, OPTICS uses core-distance and reachability-distance to store the processing order, which reflects object density; in this aspect, OPTICS is similar to AGNES, and the produced result is a hierarchical structure. Given objects o, p in dataset D, and $N_\epsilon(o)$ is the corresponding neighborhood:

- **core-distance**: the minimum neighborhood distance ϵ' for a point o to qualify as a core point. Otherwise, the core-distance is not defined.

- **reachability-distance**: the minimum distance for p to be directly density-reachable from o, when o is a core-object. If o is not a core-object, the reachability distance between o and p is not defined.

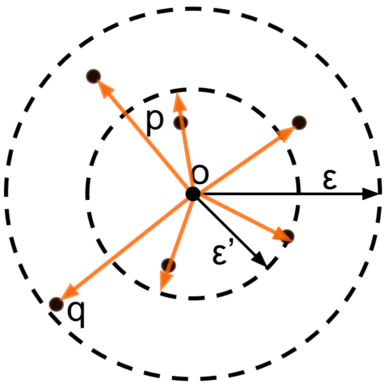


Figure 2.14: OPTICS example with $MinPts = 4$, core-distance is ϵ', and reachability-distances marked with arrows.

Fig. 2.14 shows the core-distance and reachability-distance, with core-object as o, $MinPts = 4$. Simply put, the core-distance ϵ' of o is the distance to the furthest $MinPts$ neighbor. The reachability-distance is shown in arrows, which is the greater value between the core-distance and the distance between two objects.

In essence, OPTICS is DBSCAN with a broad range of operated parameter settings. For the clustering results to be consistent, the algorithm processes a dataset in a specific order. Higher density is finished first by selecting objects that are density-reachable with the lowest ϵ. The order in which OPTICS sorts through elements in a dataset is similar to the single link AGNES process. Compared to DBSCAN, OPTICS has more relaxed parameter settings, as well as the ability to identify different densities. However, OPTICS has higher memory cost, as it maintains a priority queue to determine the next closest element to include in the current cluster.

2.3 Cluster Representation

Another important part of clustering is the representation of clusters. The mean or medoid is commonly used in this aspect. However, when dealing with time series data, it can be difficult to compute the mean or medoid, especially with lock-step measures. Without a proper time series averaging mechanism, even cluster algorithms used in conjunction with DTW does not guarantee accurate results for time series clustering [59]. Generally, averaging can be categorized as local averaging and global averaging. Here we discuss some of the time series averaging techniques.

Niennattrakul et al. [59] discussed the result using k-means with DTW. They claimed the bottleneck of clustering time series to be a lack of good averaging method for time series data. While commonly used for statistical analysis, finding the average of a time series dataset is not a trivial task. Because DTW has no triangular inequality property, calculating an average time series with the traditional Euclidean mean, even for equal length time series, would usually end with unrepresentative averaged time series.

The Non-Linear Alignment and Averaging Filters (NLAAF) [60] is where averaging is applied N times for N number of sequences. The average sequence is calculated using the center of each association, and the length can grow substantially. Prioritized Shape Averaging (PSA) [61] was proposed to deal with the shortcomings of NLAAF, PSA averages two sequences whose DTW distance is minimal to other sequences, and where each connected component is associated with a coordinate of the resulting mean. However, both NLAAF and PSA are local averaging strategies, and local averaging strategies are sensitive to order, meaning the results may change if the averaging process is repeated. It is also possible for any initial error to propagate through the entire averaging process.

A technique for robust averaging template selection called the Crossword Reference Template (CWRT) was developed to improve recognition accuracy [62]. The CWRT technique extracts a few examples from the dataset, find the sequence whose length is closest to the average length as the initial reference template. Then other templates are aligned with the template using DTW, and the final reference template is obtained by averaging the time-aligned templates across each frame. This method is invariant to the order of sequence processing; however, extracting the examples and using the closest-to-average length sequence as the initial reference template may lead to inconsistent results.

Petitjean et al. [14] proposed a global method to calculate the average of time series data, the DTW Barycenter Averaging (DBA). An initial averaging template is first chosen, then each time series sequence can be considered as a mass, with the averaging template as the orbiting center. This orbiting center is refined with DTW between each time series and initial average template by minimizing the Within Group Sum of Squares (WGSS). Given a time series set $\mathbb{S} = \{S_1, S_2, ..., S_n\}$, the time series $C = \{c_1, c_2, ..., c_t\}$ is considered an average of $\mathbb{S}$ if it minimizes:

$$WGSS(C) = \sum_{k=1}^{n} dtw(C, S_n)^2 \tag{2.12}$$

DBA is a global averaging method, which means it has no sensitivity to the order of calculation. If we could make the initialization process deterministic, then each DBA process can be repeated with no change to the result. The advantage of using DBA with DTW is that it can stretch or compress time series in an intuitive way. This is useful because it allows us to look for patterns otherwise easily missed. The two major steps in DBA computation are initialization and convergence. The initialization is where an element is chosen as a template and is a randomized choice in the original DBA algorithm. Convergence is not always smooth because DTW makes nonlinear distortions, but at each iteration, the inertia can only decrease [14]. The process ends when either the algorithm converges, or the maximum number of iterations is reached. The resulting average sequence is the same length as the initial template.

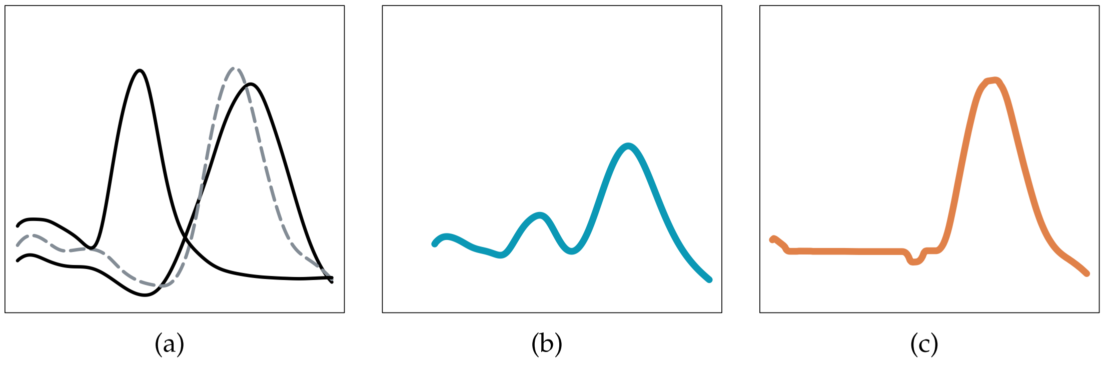


Figure 2.15: (a) Three similar, but unaligned time series, (b) average time series with Euclidean sum divided by the number of sequences, (c) averaged time series with DBA method with the second (peaking) time series as the initial template.

The effectiveness of DBA is shown in Fig. 2.15, (a) shows three similar shaped, but slightly out of phase and misaligned time series; (b) shows the average time series with Euclidean sum divided by the total number of sequences, the result is a two-peaked, flat sequence, not representative of any of the original sequences; when using the second peaking time series as the initial template, the average of the three sequences with DBA is shown in (c). Evidently, DBA provided an intuitive averaging result for time series data.

2.4 Clustering Evaluation

Being an unsupervised learning method, clustering validation is an important part of evaluating the quality of a clustering methodology. Without effective evaluation, clustering results would be difficult to be trusted or used. Generally, clustering validation indices can be categorized as internal validation and external validation. The validation indices can provide an objective measurement of clustering results. In this section, we will discuss a few commonly used evaluation indices.

2.4.1 Internal Indices

Internal validation is based on the internal information generated from the clustering process to evaluate the quality of clustering. Usually, internal validation is some form of inter-cluster and intra-cluster combination evaluation, which is used without ground truth information (data labels). Internal indices are commonly used to determine the optimum number of clusters when no such information can be obtained in prior.

Using the degree of separation and compactness, there is an abundant amount of cluster evaluation indices. The general internal validation equation is shown in Eq.2.13, it is based on a certain degree of separation compared to a certain degree of compactness of clusters. Compactness signifies intra-cluster tightness, whereas separation signifies inter-cluster differences. Here we will briefly discuss three commonly used internal indices, the Silhouette index, Dunn Index, and Davies-Bouldin index.

$$\text{Index} = \frac{\alpha * separation}{\beta * compactness} \tag{2.13}$$

Shown in Eq. 2.14, the Silhouette index (SI) has a range of [-1,1], with 1 being a perfect cluster, 0 means the observation could belong to either cluster, and negative values signify the wrong cluster. Silhouette value measures how similar an observation is to its own cluster when compared to other clusters. A SI is computed for each data element, and therefore could be expensive for larger datasets.

$$SI = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{2.14}$$

Shown in Eq. 2.15, the Dunn Index (DI) is the ratio of the smallest distance between observations belonging to different clusters to the largest intra-cluster distance. The DI value has the range between 0 and infinity; a larger DI value signifies more compact clusters. However, DI does not perform well for ring distributions.

$$DI = \frac{min\{separation\}}{max\{diameter\}} \tag{2.15}$$

Shown in Eq. 2.16, the Davies-Bouldin index (DBI) is also an internal evaluation scheme. Essentially, it is the average of the ratio between intra-cluster distance and the inter-cluster distance. DBI has a positive value, and a smaller value signifies better clustering.

$$DBI = (\frac{1}{T_i}\sum_{j=1}^{T_i} |X_j - A_i|^p)^{1/p} \tag{2.16}$$

2.4.2 External Indices

External validation utilizes results known externally, which typically refers to the ground truth. The cluster results are compared against known labels to determine cluster quality. Some commonly used external indexes include Accuracy, F-measure, RAND index, etc. Most external indexes are derived from the confusion matrix, as shown in Table 2.1.

Table 2.1: Confusion Matrix

		Actual	
		Positive	Negative
Prediction	Positive	TP	FP
	Negative	FN	TN

Accuracy performance is judged by the number of true positives and true negatives (Eq. 2.17). F-score (Eq. 2.18) is the harmonic mean of recall and precision. F-score is more resilient to class imbalance and is a better measure than accuracy when false positives and false negatives have different costs and consequences. Rand index (Eq. 2.19) measures the agreements between two data clusters.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.17}$$

$$FScore = 2 * \frac{precision * recall}{precision + recall} \tag{2.18}$$

$$RAND = \frac{a + b}{a + b + c + d} \tag{2.19}$$

Another way to evaluate the quality of clustering is by visualization. This is a qualitative method that is subjective and should only be used as an addition to other more concrete evaluation methods. Although both internal and external indexes are quantitative measures, there is no consensus on which is superior. Again, both clustering and its evaluation are relative and not absolute; it is much dependent on the data, the task on hand, and the beholder.

2.4.3 Dendrogram Evaluation

Density-based and distance-based hierarchical clustering algorithms have different concepts of similarity partitions; we want to understand their performance with time series data. The ideology behind AGNES and OPTICS are highly similar, by grouping closest objects together first and moving upwards; they are both agglomerative in structure. What makes the two clustering algorithms extremely versatile and suitable for time series data clustering, is that both can use a dataset of data elements as well as a pre-computed distance/similarity matrix. This means the user is free to incorporate any suitable similarity measure to obtain a similarity matrix that can be used for further clustering. Another advantage of the two clustering algorithms is that both have minimal parameters; this can be a great advantage when prior knowledge of a dataset is minimal. Neither AGNES nor OPTICS has an output cluster assignment, but rather a nested order of objects, which is useful in exploratory data mining.

The Cophenetic correlation coefficient (CP) [63] is commonly used in biostatistics to evaluate the quality of dendrograms. Shown in Eq. 2.20, CP is a Pearson correlation between the original distance matrix and the cophenetic distances matrix of a dendrogram. The cophenetic correlation for a dendrogram is defined as the linear correlation coefficient between the cophenetic distances obtained from the tree, and the original distances used to construct the dendrogram. Thus, it is a measure of how faithfully the tree represents the similarities among observations [64]. The cophenetic distance between two observations is represented in a dendrogram by the height of the link at which those two observations are first joined. That height is the distance between the two sub-clusters that are merged by the previous link. Therefore, CP can be used for single dendrogram evaluation, as well as comparing several dendrograms, and a higher value typically signifies better clustering.

$$CP = \frac{\sum_{i<j}(x(i,j)-\overline{x})(t(i,j)-\overline{t})}{\sqrt{[\sum_{i<j}(x(i,j)-\overline{x})^2][\sum_{i<j}(t(i,j)-\overline{t})^2]}} \tag{2.20}$$

Goodman and Kruskal's gamma measure is used in statistics for rank correlation tests for association between data points, as well as signifying how close two pairs of data points match. The Goodman and Kruskal's gamma coefficient ranges between -1 and 1, where 1 is a perfect positive correlation, 0 suggests no association, and -1 is a perfect inverse correlation. The computation is given in Eq. 2.21, N_c is the total number of concordant pairs, and N_d is the total number of discordant pairs.

$$\gamma = \frac{N_c - N_d}{N_c + N_d} \tag{2.21}$$

2.5 Datasets

In this section, we introduce three datasets associated with our experiments. The UCR Archive [21] is a general time series archive, containing distinct datasets from various fields. Whereas the ICME (Interplanetary Coronal Mass Ejections) and Solar Flare datasets are more domain-specific. The experiments show the effectiveness of our methods, and in turn, the datasets can demonstrate the wide applicability of cluster analysis.

2.5.1 UCR Archive

Most of our univariate time series work was initially tested using the UCR dataset archive [21]. At the time of this report, there are a total of 128 time series datasets in the UCR archive, including synthetic data and real-world data from various domains, ranging in class numbers as well as sizes. Each dataset consists of training and testing, and it is important to note that the training to testing ratio in the original dataset is not uniform, and the labels in training and testing portions are not always balanced. This means in application, the UCR repository is more geared towards kNN (k Nearest Neighbor) experiments, where there is no training process, and the imbalance of datasets is not a critical issue. On the other hand, the ratio of train to test, as well as the balance of labels, would greatly affect the final results for cluster analysis. Therefore, for clustering experiments, we re-sample the data by stratified sampling with five-fold cross-validation on all the datasets and only used the original split when running 1NN experiments.

2.5.2 ICME Data

Space Weather is of rising importance in scientific discipline that describes the way in which the Sun and space impact a myriad of activities down on Earth as well as the safety of the space crew members on board of the space stations. Consequently, it is imperative to better quantify the risk of future space weather events. Most solar prediction models in literature use physical parameters of the potential active regions during a limited interval to gain insights on whether an event will happen or not. This limits our perception of how an event evolves for an extended duration across multiple parameters. We approach this problem from a data-driven perspective and address the problem of prediction from a multivariate time series analysis perspective.

Different types of space weather events may have different impacts on Earth as well as to the space crew members and satellites that are in the magnetosphere [65]. For example, Solar Energetic Particles (SEP) can penetrate instruments on satellites and may be a possible threat of magnetic saturation, which can eventually lead to electrical failures. ICMEs (Interplanetary Coronal Mass Ejections) can induce extra currents in the ground, which can deteriorate power grid operations down on the Earth. Additionally, X-rays heat the Earth's outer atmosphere, causing it to expand, which can be a disturbance to satellites. As a chain effect, space weather will also impact people who rely on those technologies. In the past ICMEs (Interplanetary Coronal Mass Ejections) were believed to be caused by solar flares, later this was found to be untrue, not every flare is accompanied with ICMEs, and not every ICME happens during a flare. Due to the potential damage from solar flares and ICMEs, they are both actively studied by researchers.

Most of the studies on ICMEs published in the past primarily focused on expert-proposed parameters that are summarized from the original time series [66]. In our work, we go back to the original data recorded by the Ulysses spacecraft and focus on long-term trends in the raw time series data. We combined two datasets: (1) a list of 181 individual ICME events ranging from 1991 to 2007 [67], with labels indicating magnetic clouds (MC) and non-magnetic clouds (NMC), as shown in Fig. 2.16, the ICME events are plotted on two parameters describing the events; and (2) a multivariate time series dataset that corresponds to each event in the aforementioned event list. The labels of ICME events were provided by domain experts [67] and will serve as a benchmark for our evaluation. The multivariate time series dataset is obtained from NASA's Goddard Space Flight Center (GSFC) and contains a detailed recording from the spacecraft Ulysses that can describe the in-situ features of the ICME events in more detail. Since its launch in 1990, Ulysses has made three "fast latitude scans" of the Sun in 1994/1995, 2000/2001, and 2007/2008, exploring the three-dimensional structure of the Sun's heliosphere.

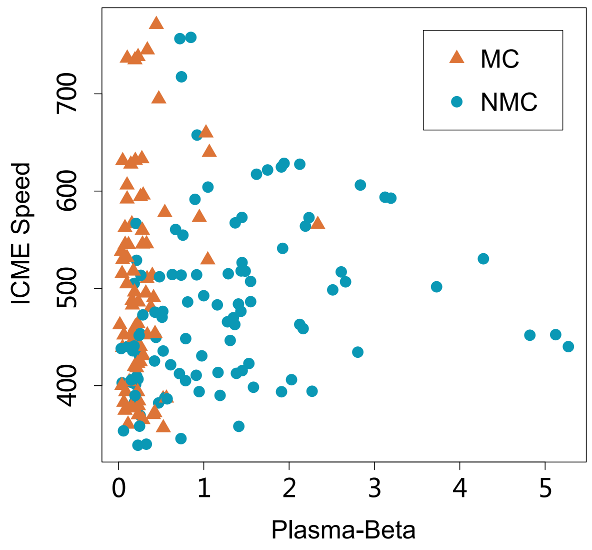


Figure 2.16: Proton Speed (ICME Speed) plotted against Plasma-Beta of the 181 ICME events used in this dataset.

The time series data from Ulysses at NASA/GSFC [68] provide us with Heliographic Inertial (HGI) coordinates, B field magnitudes, proton flow speed, proton flow elevation angle, proton flow azimuth angle, proton density, alpha density, maximum proton temperature, and minimum proton temperature. Br, Bt, Bn, and B field magnitude describe the magnetic field in the Radial-Tangential-Normal (RTN) coordinate system, and represent the in-situ magnetic field magnitude where Br is the measurement in the radial direction oriented from Sun to satellite; Bt is the measurement of the cross product of the solar rotation axis and Br; Bn is the measurement of the cross product of Br and Bt, and B is the overall magnetic field magnitude.

We also look at two additional parameters that are considered significant for event distinction [66]: (1) Plasma-Beta is a derived parameter that can be calculated with Eq. 2.22, and (2) oxygen ion charge state ratio O7/O6, which is from the "SWICS" instrument from "Ulysses" [69]. O7/O6 is a three hour averaged measurement, and all the other parameters are one hour averaged measurements. This is not an issue since all parameters are clustered independently.

$$PlasmaBeta = \frac{2n_p k_B T_p}{\frac{B^2}{\mu_0}} \tag{2.22}$$

Table 2.2: ICME Parameters

ID	Name	Description (Unit)
P01	Heliocentric Dist	Heliocentric Distance (start of data interval), AU
P02	HGI Latitude	HelioGraphic Inertial (HGI) Latitude, deg.
P03	HGI Longitude	HelioGraphic Inertial (HGI) Longitude, deg.
P04	Br	IMF Br in RTN, nT
P05	Bt	IMF Bt in RTN, nT
P06	Bn	IMF Bn in RTN, nT
P07	B	B Field Magnitude, nT
P08	Proton Speed	Proton Flow Speed, km/sec
P09	Flow Elevation	Proton Flow Elevation Angle/Latitude, (RTN), deg.
P10	Flow Azimuth	Proton Flow Azimuth Angle/Longitude (RTN), deg.
P11	Proton Density	Proton Density, n/cc
P12	Alpha Density	Alpha Density, n/cc
P13	Max Temp	Maximum Proton Temperature, K
P14	Min Temp	Minimum Proton Temperature,K
P15	Plasma Beta	Ratio of Plasma Pressure to Magnetic Pressure
P16	O7/O6	Charge State Oxygen Ratio

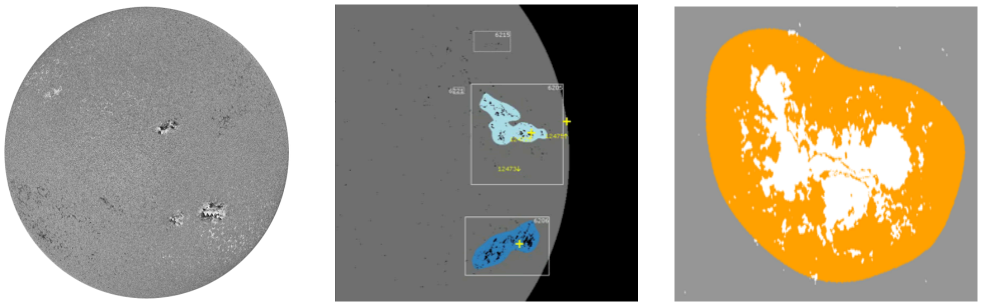
The 16 parameters are listed in Table 2.2, which includes the parameter name, and the description of each parameter. In this particular dataset, there is a small portion of invalid data in some of the earlier measurements. The longest of which only takes up a fraction of a particular event, and therefore only the invalid entries are revised, not the whole event. The domain-specific terms only refer to certain measurements and should not hinder non-solar physicists from understanding the derived results.

2.5.3 Solar Flare Data

A number of physics-based models exist for the prediction of solar flares; we took a data-driven perspective. Similar to Bobra et al. [70], we use past observations of an active region that were detected and tracked by the Space-weather HMI (Helioseismic and Magnetic Imager) Active Region Patches (SHARP) in an attempt to categorize and predict its future flaring activity.

Initial Solar Flare Data

Active regions (AR) are systematically detected and reported by the HMI team that developed a high-level data pipeline that performs this task [71]; the pipeline uses the Solar Dynamics Observatory (SDO) images shown in Fig. 2.17. Fig. 2.17 (a) is taken by the HMI instrument, which is used to detect an active region in the full-disk image data, also known as HMI Active Region Patches (HARPs), shown in Fig. 2.17 (b). A HARP is a bounding rectangle structure at the size scale of the containing solar active region. The last step corresponds to extracting the vector magnetic field maps from the HARP bitmaps, as shown in Fig. 2.17 (c). The latter maps along with other magnetic field physical properties, and are called Space-weather HMI Active Region Patches (SHARP). The 16 parameters are shown in Table 2.3. The HMI data repository is made accessible by the Joint Science Operations Center, which provides continuous measurements of the magnetic field taken from space. Here, the non-flare class is made of active regions that never led to any flare with intensity class greater than class B.



(a) (b) (c)

Figure 2.17: Illustration of the feature extraction phases steps. (a) Shows an illustration of an image taken on board of the SDO which shows the vector magnetic field that is further used to detect the active regions in (b) that are masked in (c) before computing the magnetic field related features. [1]

Our goal is to determine which active regions would lead to a solar flare (flare class), and which active regions did not lead to any flares (non-flare class). The flare catalog by NOAA is consulted to find the parent active regions where the flare was initiated. As soon as a solar flare is detected from the GOES X-ray Sensor instrument (XRS), it is reported to the flare catalog and is usually paired with its parent active region. The NOAA flare catalog is a widely accepted catalog due to the continuity of the GOES missions, which started in 1974 with the launch of SMS-1 satellite and continued to today with the GOES-15. There are five categories of flares that are detected by the GOES satellites. The less invasive ones are class A and B, whose X-ray flux values are lesser than 10^{-6}. The non-flare class is active regions that did not lead to any flares during its lifetime to a class C or higher. Class C, M and X flares are the ones whose X-ray flux values are greater than 10^{-6}, 10^{-5} and 10^{-4} $Watts/m^2$, respectively [70], and are considered as flares in our analysis.

We extracted six datasets that correspond to different observation periods and prior periods. The *prior* signifies the number of hours prior to the flare event occurrence. The prior can be thought of as being an interval of time in the future when our model

Table 2.3: Active Region Parameters

ID	Tag	Description
P01	USFLUX	Total unsigned flux
P02	MEANGAM	Mean angle of field from radial
P03	MEANGBT	Mean gradient of total field
P04	MEANGBZ	Mean gradient of vertical field
P05	MEANGBH	Mean gradient of horizontal field
P06	MEANJZD	Mean vertical current density
P07	TOTUSJZ	Total unsigned vertical current
P08	MEANALP	Mean characteristic twist parameter α
P09	MEANJZH	Mean current helicity (B_z contribution)
P10	TOTUSJH	Total unsigned current helicity
P11	ABSNJZH	Absolute value of the net current helicity
P12	SAVNCPP	Sum of the modulus of the net current per polarity
P13	MEANPOT	Mean photospheric magnetic free energy
P14	TOTPOT	Total photospheric magnetic free energy density
P15	MEANSHR	Mean shear angle
P16	SHRGT45	Fraction of Area with Shear $>$ 45 deg

should be able to predict the occurrence or non-occurrence of a flare. We considered two categories of prior periods, which are 12 and 24 hours. In other words, we investigated the possibility of predicting a flare from an active region's physical characteristics 12 or 24 hours before their potential occurrence. The *span* signifies the observation period; in other words, the number of hours we observe the said active region. There is a direct correlation between the span duration and the length of the time series. Here, we selected three different spans: 6, 12, and 24 hours and two different priors: 12 and 24, as shown in Table 2.4.

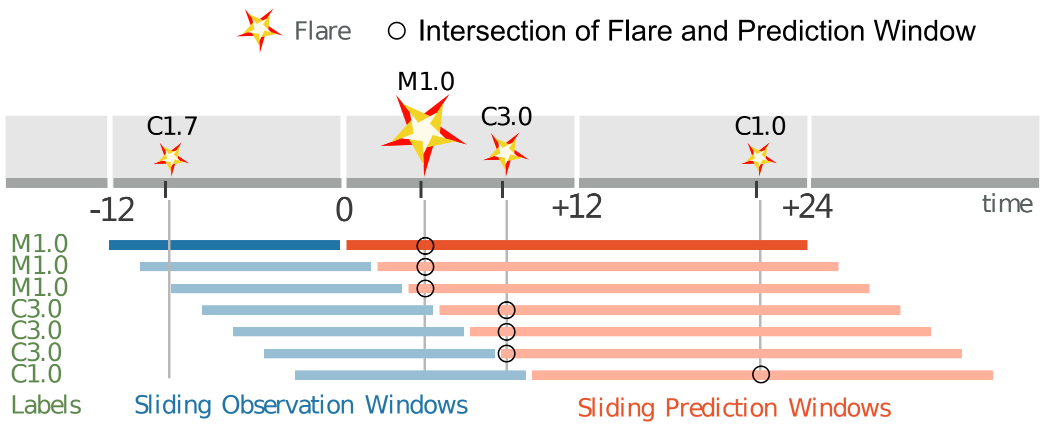
Table 2.4: Solar flare datasets with prior $\in \{12, 24\}$ and span $\in \{6, 12, 24\}$

	Dataset Type	Time Period	Data		
			Span 6	Span 12	Span 24
Prior 12	Train	2011,2012,2013,2014	6924	6446	5683
	Test	2015,2016	2383	2185	1818
Prior 24	Train	2011,2012,2013,2014	6743	6279	5553
	Test	2015,2016	2283	2056	1741

In order to validate our model, splitting the data into a training set and a testing set is required. The most common sampling methodologies include holdout and *k*-fold cross-validation. Here, we used a temporal split since we are dealing with solar flares where the temporal dimension is important due to the fluctuation of Sun activity, which depends on the solar cycle. The goal of the temporal split is to ensure the robustness of the model by making sure there is no dependency on the Sun's activity level in a particular time period. To ensure the validity and applicability, when using this dataset, not only are we testing on never-seen data but also data that is temporally non-overlapping with the training data time.

Extended Solar Flare Data

For time series cluster profiling, we use the Space Weather ANalytics for Solar Flares (SWAN-SF) [72], which is a benchmark dataset of multivariate time series (MVTS), spanning over the 9-year period (2010-2018) within the solar cycle 24. The SWAN-SF solar flare dataset is more comprehensive than the initial solar flare dataset. The time series in the data, as illustrated in Fig. 2.18, are the result of a sliding temporal window spanning over 12 hours of observation prior to the occurrence of an event. The sliding window extracts a list of physical (magnetic field) parameters from regions of interest (RoI) and produces an MVTS. The label assigned to each MVTS corresponds to the strongest flare in the temporal observation window, among the multiple flares that may co-occur. Of course, the presence of multiple flares in an observation window impacts the general flares' profiles. However, an operation-ready forecast system also needs to predict based on the characteristics of flare clusters, and not singled-out flares. This is simply a design choice in SWAN-SF, made to closely mimic the data that any real-time forecasting model should eventually base their predictions on. If no flares were reported during that period, the corresponding MVTS would be labeled as flare-quiet (FQ). Note that each MVTS is unique to a particular active region. However, due to the use of a sliding window for slicing time series, multiple observation windows may be attributed to a single flare occurring in that region. This behavior is reflected in Fig. 2.18 by the top-tree blue bars, representing three different MVTS, all corresponding to one flare of magnitude M1.0.



Flare
Intersection of Flare and Prediction Window
M1.0
C1.7
C3.0
C1.0
-12
0
+12
+24
time
M1.0
M1.0
M1.0
C3.0
C3.0
C3.0
C1.0
Labels
Sliding Observation Windows
Sliding Prediction Windows

Figure 2.18: The slicing process for the SWAN-SF benchmark dataset. For a given active region, each observation window (blue bars) is labeled (on the left, in green) according to the intersection (black circles) of its prediction window (red bars) with the magnitude of the largest flare reported for this active region in that time window. Note that each observation window produces one MVTS.

This data benchmark comprises of five partitions in such a way that there is approximately an equal number of strong (GOES M- and X-class) flares, distributed in each partition. The partitions are temporally separated. This provides an easy way for users to split the data into training, validation, and testing sets, without having to worry about unwanted biases that their sampling methodology may impose on the problem. Table. 2.5 shows the number of instances (pre-flare time series) labeled in compliance with the settings of Fig. 2.18: a 12-hour observation window and a 24-hour forecast window, with zero latency.

Our analysis is based on the notion that flare data do not only differ in value but also in the development process, and it is the latter that is our emphasis. As this dataset is imbalanced and our clustering requires balanced datasets, we did not want to emphasize the quantity of experimental data. Instead, we selected 100 unique C- and M-class instances from Partitions 4 and 5, respectively. Moreover, because X-class instances are

Table 2.5: Labeled flare instances as per flare class for the five partitions of the SWAN-SF benchmark dataset. Also shown is the approximate class imbalance ratio between labels corresponding to M- and X-class flares and all the other labels.

Partition	X	M	C	B	FQ	Ratio
1	172	1130	6250	4999	64222	1:58
2	48	1279	8444	4194	78517	1:69
3	160	1152	3350	108	22236	1:20
4	165	1153	6487	832	52689	1:51
5	21	1071	6419	832	89400	1:95

rare events, and it is important to have balanced datasets for cluster analysis, we selected X-class instances from across all five Partitions. For obtaining 100 X-class flare instances, if they have similar active region id, we limited our sample space to time series that are as spread out as possible. This is to avoid the impact of auto-correlation caused by the time series coming from the same active regions.

3 SEGMENTED DYNAMIC TIME WARPING (SEGDTW)

As introduced in Section 2.1, while DTW is useful in comparing sequential data, it has two major flaws: high computational cost and the possibility of pathological warping paths. The origin of the two flaws is the linear calculation of mapping of two sequences [73]. The warping path, or shortest global route, for DTW is identified based on the computation and comparisons of several options at each step. While this contributes to the general success of the DTW algorithm's performance with time series data, the large-scale computations make DTW a very time-consuming algorithm. The other issue with DTW is the occurrence of pathological warping path, which happens when the warping path tries to compensate for small value differences. In other words, pathological warping paths exist because the decision at each step does not take significant global similarities into consideration. When the path is found in a greedy manner and never readjusted, the optimal warping path could be permanently missed.

In application, we can often observe cases where the warping path between two time series does not match the expected intuitive mapping, to solve this problem, different constraints are applied. However, the tightness of global constraints for DTW is an issue not often discussed. For efficiency improvements, when the window constraint becomes too tight, the overall performance can be reduced, and in the most extreme case, DTW becomes Euclidean distance. When dimension of time series is reduced to much to speed up time series computation, the significance of using time series data could be dissipated. We approach this problem in a two-step approach, global similarity can be identified by adding a layer of significant feature identification, which acts as a form of restraints;

then the local similarity can be identified by focusing on each corresponding segments individually [74].

Fig. 3.1 shows the mapping comparison between the standard DTW algorithm and an example of segDTW with peaks. In this example, the standard DTW mapping is not matching our intuitive expectations and is showing signs of pathological warpings. In contrast, by identifying two peaks and segmenting the two time series into three paired segments, the segDTW mapping provides more intuitive results. When features are identified and paired, not only do we obtain a more intuitive mapping, but also makes it possible and exceptionally easy to parallelize DTW computation within a time series sequence. Given enough computing resources, we can greatly improve the time cost of distance matrix computation, the segmentation makes our method scalable and sets segDTW apart from other DTW improvement heuristics.

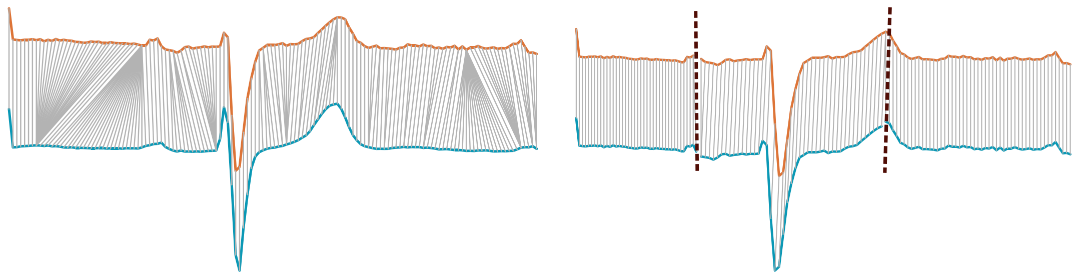

(a) Standard DTW mapping (b) segDTW mapping

Figure 3.1: A comparison between the standard DTW and segDTW, (a) is the mapping of standard DTW, which is computed linearly and is showing signs of a pathological warping path, (b) shows the mapping results of segDTW with peak identification, here each segment can be simultaneously computed. Hence, segDTW can generate faster and more intuitive results.

This chapter presents Segmented Dynamic Time Warping (segDTW), which differs from previous DTW improvement works in the sense that it is designed to break the linear barrier and take the DTW computation to parallel. Section 3.1 discuss how global identifying features are established, and Section 3.2 explains how the features are paired. Through detecting features of time series data and segmenting accordingly, a layer of approximation is added prior to DTW distance computation. While this idea is straightforward, it is extremely effective in providing scalability and improving speed.

3.1 Feature Selection

For feature selection, any meaningful and easy to identify features can be used, in our work we proposed a simple yet flexible peak/valley detection heuristic. The naïve definition of a peak can be formulated as follows: the temporal index i corresponds to the peak c_i, if $c_i > c_{i-1}$ and $c_i > c_{i+1}$, and c_i is referred to as *candidate peaks*. Similarly, a valley can be formulated as the temporal index i corresponds to the valley c_i, if $c_i < c_{i-1}$ and $c_i < c_{i+1}$, and c_i is referred to as *candidate valley*. This simple definition equipped with peak/valley selection parameters t, d, and n form a peak/valley detection method that provides the criteria necessary to distinguish a set of selected peak/valley used in segDTW referred to as *significant peaks/valleys*.

- t: threshold on the frequency domain. Any candidate peaks below t will not be considered for peak detection, and any candidate valley above t will not be considered for valley detection.
- d: the minimum peak/valley radius. For any identified peak/valley, any adjacent peak/valley within a radius of d will be considered as either noise or insignificant.
- n: the maximum number of features to be taken into account. Since the values of features are sorted before being analyzed, only the top n features that have met the other criteria will be considered.

Being a variant of DTW, the basic conditions of boundary, monotonicity, and continuity are also applicable for segDTW. As an effect of the boundary condition for segDTW computation, the end points for each segment are specifically mapped. Next, the optimal

DTW mapping is found for each sub-sequence, and the distances of all the sub-sequences are aggregated to obtain the overall distance between two time series sequences. This method can alleviate singularities, minimize mismatches, and avoid pathological warping paths.

For simplicity, Algorithm 3.1 only shows the selection for feature peaks, but the conditions can be simply changed according to the previous description for valley detection. Initially, all candidate peaks are found and sorted based on their values (Algorithm 3.1 lines 3-7). Then, the threshold t on the frequency domain is applied, and all the candidate peaks below the threshold will be removed from the list (lines 8-13). For the remaining peaks, the neighboring peaks within the radius of d temporal indices will then be removed (lines 15-21). The algorithm processes the peaks in a top-down fashion, as this can guarantee that the presence of a smaller peak will never justify the removal of a more significant peak. Finally, the top n peaks can be optionally picked among the remaining peaks on the list.

3.2 Feature Pairing

Once the features are identified, we need to pair the detected features. The mapping method in such a situation plays a crucial role in achieving sensible results. A prerequisite condition for segDTW is that it has to satisfy all the basic requirements of DTW, namely boundary, monotonicity, and continuity conditions. This can be extended to the condition that no set of paired features in segDTW could cross another pair of features, as this would be a violation of the monotonicity requirement of DTW. Since segDTW consists of standard DTW, when boundary and continuity requirements are satisfied for each time series segment, they are automatically satisfied as a whole.

Algorithm 3.1 Time Series Peak Selection

Input: $c = \{c_1, \cdots, c_m\}$ time series data,
t: the minimum threshold on the frequency domain below which all peaks are ignored,
d: the minimum radius from a selected peak within which all other peaks are ignored,
n: the maximum number of peaks to be detected.
Output:
the list of peaks with both their indices and values.

```
1: procedure FIND PEAKS
2:     $significant.peaks, candidates \leftarrow$ list()
3:     for all $c_i \in C$ do
4:         if $((c_i > c_{i+1})$ & $(c_i > c_{i-1}))$ then
5:             $candidates$.add(($i$, $c_i$))
6:         end if
7:     end for
8:     $peaks \leftarrow$ sortByValue($candidates$)
9:     for all $(i, c_i) \in peaks$ do
10:        if $c_i < t$ then
11:            $peaks$.remove(($i, c_i$))
12:        end if
13:    end for
14:    $indices \leftarrow peaks$.getIndices()
15:    for all $i \in indices$ do
16:        for all $j \in \{i - d, \cdots, i + d\} \setminus i$ do
17:            if $peaks$.hasIndex(j) then
18:                $peaks$.remove((j, $c_j$))
19:            end if
20:        end for
21:    end for
22:    $significant.peaks \leftarrow peaks$.getNFirstElements($n$)
23:    return $significant.peaks$;
24: end procedure
```

The mapping of features, in many situations, is a subjective task, and it is not always possible to agree on any ground truth even when visually analyzing the time series. Therefore, we set our goal to minimize the total distances of the pairs. To this end, we employ the Hungarian algorithm [75] in order to achieve a global optimum feature pairing. This is a generalized choice, depending on the situation, this choice can be alternated with any other user-preferred method that potentially suits the dataset or task at hand.

The Hungarian algorithm is a well-known assignment problem which is regarded as a relative of the traveling salesman problem. The assignment problem can be formulated as follows: given an n by n matrix $R = (r_{ij})$ of non-negative integers, the objective is to find the permutation $\{j_1, j_2, \cdots, j_n\}$ of the integers $\{1, 2, \cdots, n\}$ such that it minimizes the sum $r_{ij_1} + r_{ij_2} + \cdots + r_{ij_n}$. The Hungarian method utilizes linear programming to tackle this problem. Although the assignment problem can always be reduced to the case where R takes only ones and zeros, representing paired and non-paired integers, it is not limited to this binary case. Also, the problem can be easily extended to the non-square matrices. Thus, this approach is a natural choice and is well suited for our feature-pairing problem. The sequence of the integers, in this context, is the temporal indices of the features in the two time series, Q, and C, and the permutation of interest is the mapping between the features from one time series to the other. The entries of the assignment matrix R for our problem can be defined as follows:

$$r_{ij} = \text{temporal distance between } \hat{q}_i \text{ and } \hat{c}_j$$

where $\hat{q}_i$ and $\hat{c}_j$ are the i-th and j-th feature in the time series Q and C, respectively. The objective here is to find a permutation that minimizes the total temporal distance of the features.

Two important adjustments to the original Hungarian algorithm need to be made to fully customize this solution to fit the task of pairing significant features. Our main objective here in adjusting the feature pairing method is to mimic the human decision-making process. First, the algorithm must avoid situations where two pairs of features are mapped in a reverse temporal order. The mapping of one pair crossing the mapping of another pair of features is never allowed, as this would be a violation of the monotonicity requirement of DTW. Coincidentally, this also ensures the algorithm to drop the more costly feature mappings. Second, When the number of significant features is different in the two time series, which occurs more often than not, only those contributing to the global optimum are selected. The pairing of features is high-level decisions that must be made even when a practitioner is assigned to tackle subjective tasks of this kind. In most cases, the most optimal decision is the one that aligns with our intuition.

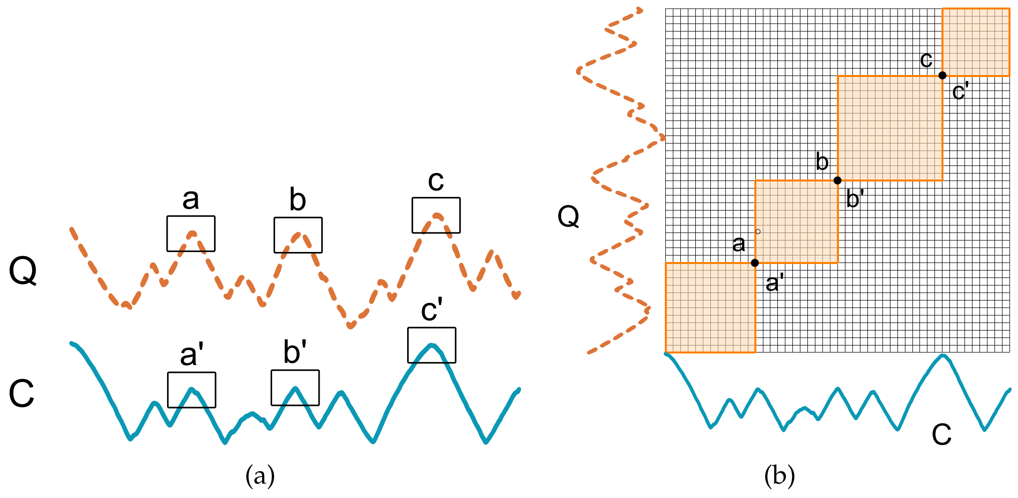


Figure 3.2: (a) Standard DTW mapping with three pairs of peaks identified. (b) Dissimilarity matrix used for segDTW with identified peaks, the warping path can only traverse through the colored dynamic sub-windows.

Another way to understand segDTW is to consider it as a form of *dynamic windows* [23]. Shown in Fig. 3.2 (a), when the three identified peaks a, b, and c are mapped to a', b', and c' respectively, they correspond to the three dots in Fig. 3.2 (b). The paired peaks define the colored squares, which can be treated as dynamic sub-windows for DTW. The warping path would have to traverse through the points of mapped peaks. Any mappings outside of these sub-windows do not need to be computed, and this is extremely important since this not only speed up the computation, but also provides more intuitive results. The computations pruned off by the segDTW model is not done for the sole purpose of achieving a lower computational cost, but more importantly, the choice is justified by intuitive mapping. Based on the behavior of time series sequences, the detected global similarities place a coarse constraint on the overall warping path.

Compared to global constraints, segDTW allows more freedom for a warping path to develop within each sub-window (segment pair).

In the worst scenario, the time complexity of segDTW, if implemented as described, is $max\{O((p \cdot (p+1))/2), s)\}$ where p is the number of candidate peaks/valleys and s is the time bound of the utilized sorting algorithm. The worst case refers to the situation where $d = 0$, $n = \infty$, and $t = min(C)$. However, by taking the order of the indices into account, in addition to the order of the values, the time complexity would only be determined by the sorting step. Hence, the worst-case running time would be decreased to $O(n \cdot \log(n))$, which reflects the complexity of a sort algorithm such as merge sort or heap sort.

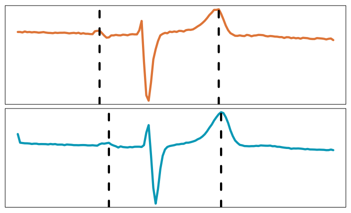
3.3 segDTW Performance

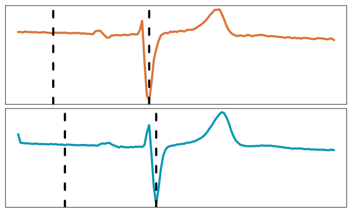
We demonstrate our findings with datasets from the UCR time series archive [21]. The original train-test split is used with this 1NN classification. Although the time series length is uniform across each dataset, segDTW can be applied toward time series of different lengths just as DTW can. All the measurements, DTW, DDTW, as well as segDTW, are used as is, meaning no further optimization methods applied. This holdback means we purposefully slowed down the experiments to ensure fair comparisons. There is no parallelization on the time series event level for any of our experiments. When using any similarity measurements, the computation between each training and each testing event can always be distributed to different cores. In other words, the parallel computation of time series events can be done for any DTW based method. Therefore, we take away the common factor and utilize multiple processor cores for computation only when time series are segmented. Parallelization occurs only within sequences, and never between sequences. In short, our experiments are overall slowed to a controlled state of having one single variable: segmentation.

3.3.1 Feature Type

We first explore the importance of significant features by comparing global features peak and valley with equal window segmentation. Instinctively, by segmenting one computation into several pieces and processing each piece simultaneously should improve the overall time cost. However, in order to obtain meaningful results in sequential comparison, the segmentation would also have to be meaningful. While meaningful cannot be precisely defined, it does insinuate that the segmentation corresponds to intuitive human interpretation.

Fig. 3.3 shows the segDTW with peaks as features, valleys as features, and also the same number of equal-width segments. The two time series in Fig. 3.3 (a) are divided into three segments based on the two peaks identified as significant features, and the accumulated segDTW distance is 16.16. One significant valley and one miniature valley are found in Fig. 3.3 (b), and the accumulated segDTW distance is 18.75. When time series are segmented into equal-width segments, as is the case in Fig. 3.3 (c), the accumulated segDTW distance is 22.14. Although the processing time is approximately the same for the three cases, the segmentation for the equal-width window does not have any actual meaning. This situation could be worsened as the number of equal-width segments grows, and in the extreme case for equal length time series, when each window has a width of 1, segDTW with equal-width segment degenerates to the L_p-norm Euclidean distance.


(a) Peak as feature


(b) Valley as feature

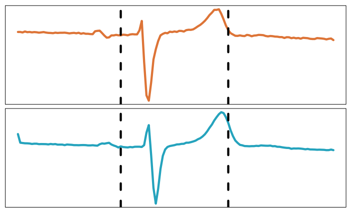

(c) Equal-width window

Figure 3.3: Two time series computed with segDTW with (a) peak, with distance 16.16 (b) valley, with distance 18.75, and (c) equal-window, with distance 22.14, respectively.

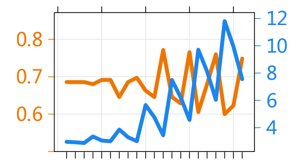
Both peaks and valleys are easily distinguished and can be used for sequence segmentation; either could be more effective depending on the dataset and application. For our experiments, we are solely using peaks as it is easy to recognize, simple to implement, and is not seemingly performing worse than using valleys.

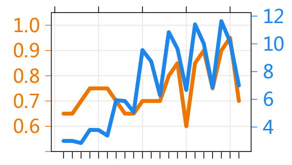
3.3.2 Peak Selection Parameter Effect on Performance

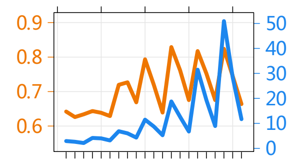
Generally, when the peak selection radius d is large or when the peak threshold t is large, fewer peaks are identified, which means fewer sub-sequences. In the extreme case, when no peaks are detected, and no segmentation is imposed, segDTW simply becomes the standard DTW. In contrast, when the peak detection radius is small or when the peak threshold is low, there are more peaks and sub-sequences, which leads to more segments and potentially shorter processing time. However, more segments in time series sequences introduce risks of identifying false peaks, which could lead to poor performance.

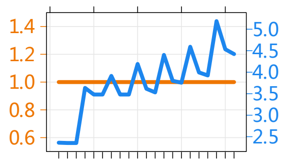
While any segmentation of time series would improve computation efficiency, it is the quality of segmentation and its associated mapping that is our focus. Although general parameters can help us gain some insight into the data, for optimum performance, specialized parameter settings should be applied for specific datasets.

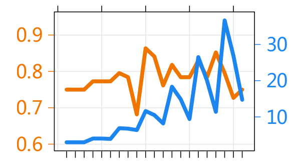
The number of identified features change when different segmentation parameters are applied, and it is not necessarily a linear change. Fig. 3.4 shows how the number of peaks and its corresponding accuracy change with different peak selection parameter settings. The horizontal axis is the permutations of parameter settings d=$\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{32}$, $\frac{1}{64}$ and $\frac{1}{128}$ of data length L, and t with values of first quantile (Q1), median (M) and the third quantile (Q3); here no n imposed. For simplicity, parameter d is labeled with the denominator in Fig. 3.4. The left-y-axis is the accuracy, and the right-y-axis is the average number of paired peaks. The orange line corresponds to the accuracy, and the blue line corresponds to the number of peaks. In order to show the detailed accuracy

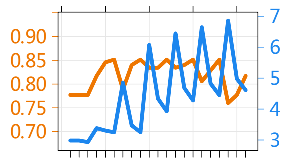

(a) ArrowHead

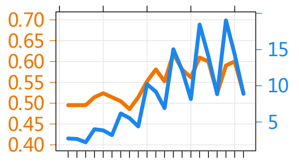

(b) BeetleFly

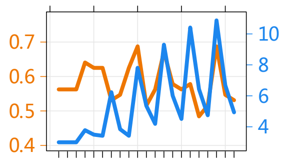

(c) CinC_ECG_torso

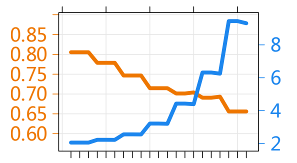

(d) DiatomSizeReduc

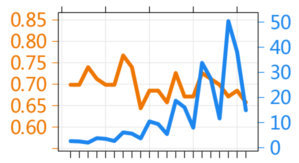

(e) FaceFour

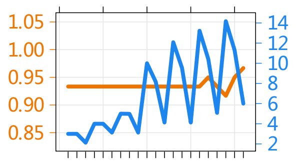

(f) Fish

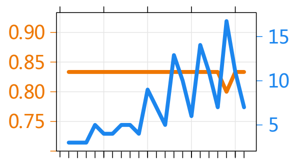

(g) Ham

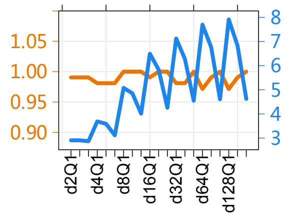

(h) Herring

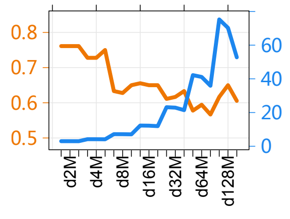

(i) LgKitchenApp

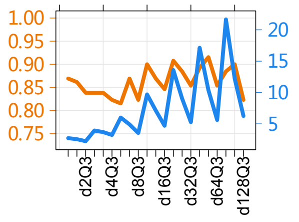

(j) Lighting7


(k) Meat


(l) OliveOil


(m) Plane


(n) ShapeletSim


(o) ToeSeg2

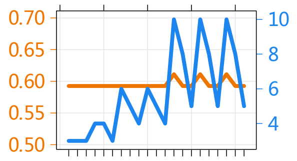
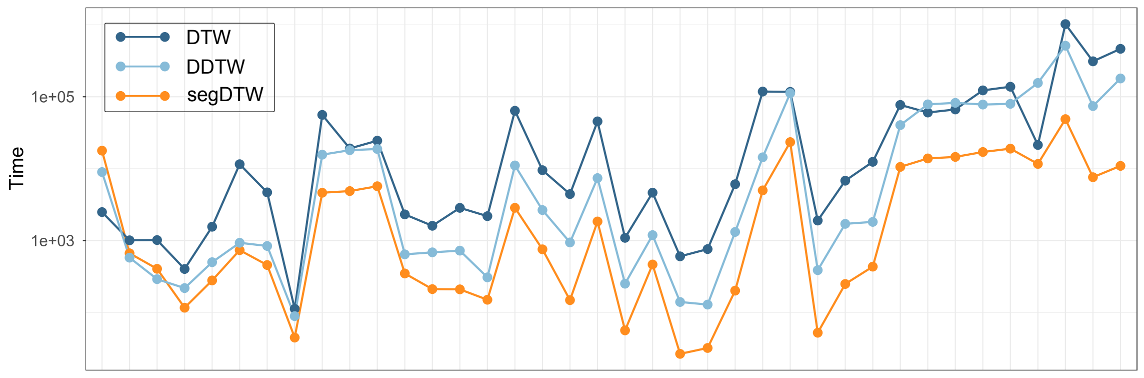

(p) Wine

Figure 3.4: The relationship between the number of peaks with accuracy for 16 datasets.

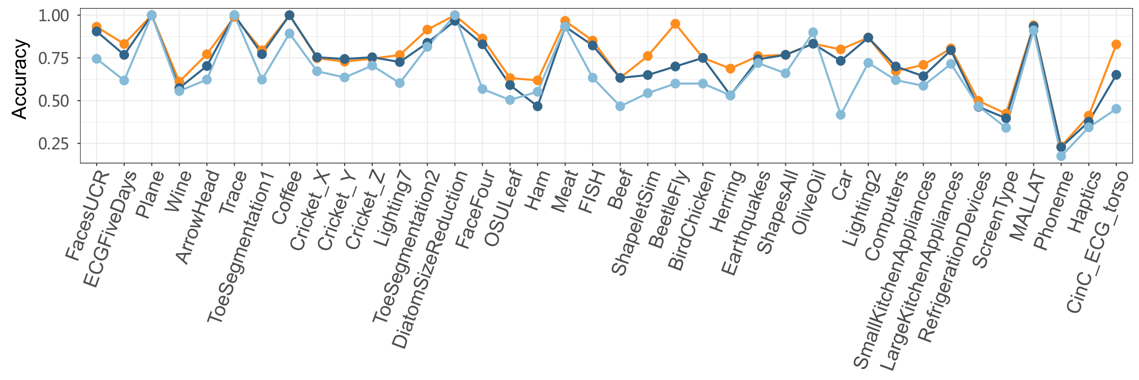
fluctuation, we did not use a uniform accuracy range, and the accuracy range for each dataset is varied. While the results are distinct from dataset to dataset, there are some identifiable trends. For datasets shown in (d), (k), (l), and (p), accuracy does not fluctuate significantly in relation to the number of peaks. The relationship between accuracy and number of peaks is positive in ratio for datasets (b), (c), and (g). While the relationship between accuracy and the number of peaks is inverse in ratio for datasets (i) and (n). For some datasets, the maximum accuracy occurs for a low number of paired peaks, while others occur specifically for a high number of paired peaks. In short, harmonious to the no free lunch (NFL) theorem, parameter setting, and the quality of paired features differ depending on the dataset.

3.3.3 segDTW Efficiency

For efficiency and accuracy performance, we used 38 datasets from the UCR archive and compared three similarity measurements using 1NN. Fig. 3.5 (a) shows the computation time, and (b) shows the accuracy for standard DTW, DDTW, and segDTW, with datasets on the horizontal-axis ordered by their length in ascending order. The general processing time varies drastically from dataset to dataset caused by the difference in data length; therefore, the time-scale is depicted on the logarithmic scale. Generally, DDTW is more efficient than standard DTW, and segDTW is more efficient than DDTW. Out of the 38 datasets, there is only 1 case where segDTW did not outperform the standard DTW and 2 cases where segDTW did not outperform DDTW, all of which occurred for the three datasets with the shortest data length. There are also 19 cases where compared to the standard DTW, segDTW had over ten-times increase in processing time. Since the scalability of segDTW is segment dependent, it makes sense for this advantage to be more apparent for datasets with longer time series sequences; when the overhead of the time spent on segmentation becomes less relevant compared to the linear calculation time spent on an entire sequence pair. Among the datasets with the longest sequences, we saw an increase of over 40-times in computation speed for 2 datasets. In the optimal situation, segDTW performed the same or better in accuracy for 31 out of 38 datasets, without using any other DTW improvement methods.



(a) Time cost in logarithmic scale.



(b) Corresponding accuracy performance

Figure 3.5: Datasets are in ascending order of time series length, (a) shows the processing time for standard DTW, DDTW, and segDTW on the logarithmic scale, with segDTW speedup becoming more apparent as time series length increases; (b) shows the corresponding accuracy performance.

3.3.4 segDTW Application

Another advantage of segDTW is that it does not have to be a stand-alone heuristic. Due to its unique divide-and-compute structure, segDTW can be combined with almost any existing DTW improvement approach. Here we will demonstrate segDTW combined with a global warping window. Table 3.1 shows the performance of segDTW with the Itakura warping window compared to the standard DTW and segDTW. In general, segDTW combined with an additional warping window is more efficient than the segDTW alone.

However, there is a slight decrease in the accuracy of the stacked procedure; this can be attributed to the tightness of constraint, as there is always a rist of over-contraining the warping path when we pursue efficiency.

Table 3.1: Time and Accuracy for DTW, segDTW, and the Combination of segDTW with the Itakura Window

Dataset	DTW Time	DTW Accuracy	segDTW Time	segDTW Accuracy	segDTW & Itakura Time	segDTW & Itakura Accuracy
DiatomSizeRed	2851.152	0.967	209.814	1.000	200.943	0.938
FaceFour	2184.636	0.830	150.442	0.773	140.988	0.761
Ham	9517.044	0.467	755.781	0.581	755.268	0.543
Lighting7	2313.384	0.726	348.706	0.712	324.118	0.721
Meat	4426.224	0.933	148.530	0.933	133.588	0.933
ToeSeg2	1606.704	0.838	211.372	0.877	215.616	0.631

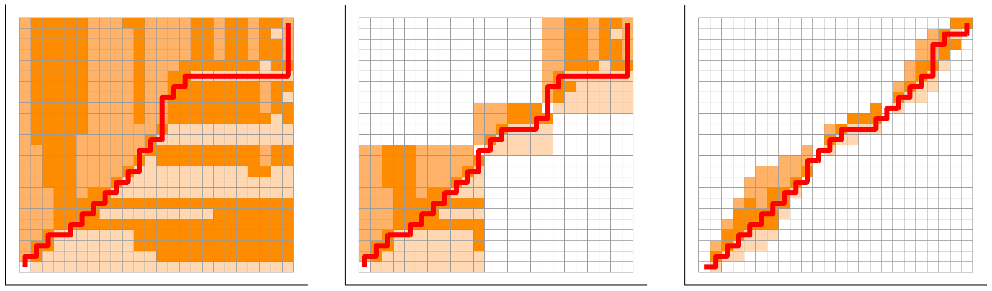

(a) Standard DTW. (b) segDTW. (c) segDTW with window.

Figure 3.6: Warping path of (a) standard DTW, (b) segDTW, and (c) segDTW combined with the Itakura warping window

For a more visual understanding, the warping path for the standard DTW is shown in Fig. 3.6 (a), segDTW is shown in Fig. 3.6 (b), and segDTW combined with the Itakura warping window is shown in Fig. 3.6 (c). Each orange cell corresponds to a pairwise computation. In this example, segDTW identifies two peaks, and the white area outside the dynamic sub-windows are pruned. segDTW with the Itakura warping window has the least amount of computation. The warping paths are drawn in red, and compared to the standard DTW, segDTW avoided some pathological warpings. While segDTW combined with the Itakura warping window provides a very tight constraint for the warping path.

In practice, segDTW would also work with methods such as slope weighting, step-pattern, lower bounding, approximation, shape descriptor, early abandonment, and more.

The trade-off between efficiency and performance can be observed with many DTW improvement methods. With little or no restraints, the warping path may stray pathologically, which also adds to the computation cost of the similarity matrix. When the restraints are too tight, we risk missing the optimal warping path, threatening on Euclidean distance, and possibly negating the advantages of time series data usage.

4 DISTANCE DENSITY CLUSTERING

Distance Density Clustering algorithm [76, 77] is a medoid-based clustering algorithm combined with virtual density regions that constitute natural splits within a dataset. We use Dynamic Time Warping and DTW Barycenter Averaging as a means to compare and represent time series data. Our cluster heuristic is formulated as a divisive hierarchical structure, which is useful for visualization and incremental clustering. Section 4.1 introduces how DDC identifies the initial clustering seed, and by deterministic initialization, we can guarantee good and stable clustering result that is reproducible. Section 4.2 demonstrates how the clustering is conducted with each iteration. By considering the distribution of the data, we are able to incorporate the ideologies of both distance-based clustering as well as density-based clustering.

4.1 Clustering Initialization

Cluster initialization is generally the very first step for a partitioning clustering algorithm. Initialization is important as it determines if the clustering algorithm is reproducible as well as affecting the convergence and performance. Since we should only expect local optimum results, we explore the impact of deterministic initialization. This issue has been addressed with *k-means++* [58]. The intuition behind k-means++ is the belief that spreading out the initial seeds is good, and in the case of outliers, the algorithm would readjust the clustering seeds.

Inspired by the k-means++ algorithm, we approached the initialization process from two perspectives, starting from the perimeter (less dense) region and moving in, and starting from a dense region and moving out. In other words, we use the nearest and furthest point as the initial seed and would also compare to the case with randomly selected seed. By selecting the first seed as a specific point, the clustering process would become a deterministic and reproducible process. In order to select the furthest or nearest point, we apply a majority voting system.

Table 4.1: Similarity Matrix for Five Time Series

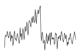
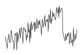
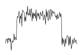
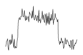
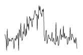
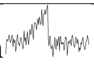
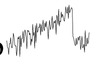
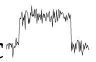
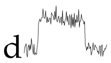
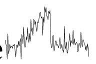
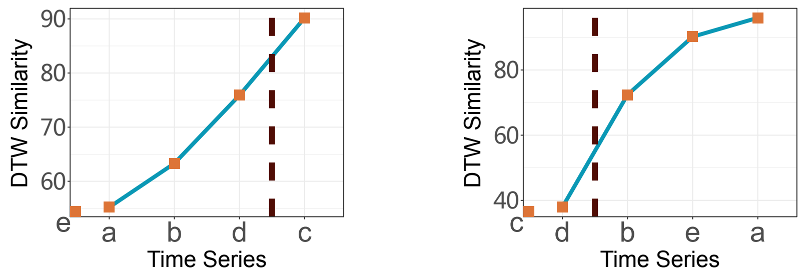
	 a	 b	 c	 d	 e
a 	0	66.2	96.0	81.6	55.2
b 	66.2	0	72.4	69.2	63.3
c 	96.0	72.4	0	37.9	90.2
d 	81.6	69.2	37.9	0	75.9
e 	55.2	63.3	90.2	75.9	0
Nearest	1	0	1	1	2
Furthest	2	0	3	0	0

Table 4.1 shows an example of a distance matrix of five time series, from top to bottom and left to right, the time series are referred to as *a*, *b*, *c*, *d*, and *e*. From the original dataset class label, there are two classes within the five time series data, which are $\{c, d\}$ and $\{a, b, e\}$. Our majority voting strategy is simple, the nearest seed is identified as the time series that is the most similar to the most number of other time series, whereas the furthest seed is the time series that is the most dissimilar to the most number of other time series. In this example, the nearest time series is event *e* with 2 votes, and the furthest is *c* with 3 votes.

After obtaining the initial seed, we need to identify a good place to separate the dataset and to also search for the next seed. We do this by plotting a similarity plot of the seed we have to all the other time series sequences within that dataset/cluster. The elbow point, meaning where the plot shows the strongest bend, corresponds to a virtually sparse region in the group of time series events, which provides a good place to split the dataset/cluster. This process is similar to finding the ϵ value (maximum radius of the neighborhood from cluster seed) in DBSCAN. Fig.4.1 shows the distance plot for (a) the nearest time series seed e, and (b) the furthest time series seed c. The difference is that while we can easily visualize a dense or sparse region within a discrete dataset with DBSCAN, it is near impossible to visualize the density within a time series dataset; therefore, we use time series similarity separation to mimic virtual sparsity.



(a) Similarity plot of nearest seed 'e' (b) Similarity plot of furthest seed 'c'

Figure 4.1: Similarity plot of (a) nearest time series seed and (b) furthest time series seed. Visually, the strongest bend is the steepest slope, which occurs between time series 'd' and 'c' in (a), and between time series 'd' and 'b' in (b). This bend is conceptually a virtually sparse region, and a suitable space to partition the dataset/cluster.

By definition, clustering is where within-cluster distances are minimized and between cluster distances maximized; or in other words, locating the regions of high density that are separated from one another by regions of low density. Therefore, by plotting time series based on their similarity to the seed in ascending order and identifying the region in the plot where the bend is the strongest, we can identify a more natural separation. Essentially, a strong bend in a similarity plot signifies a virtual sparse region within the time series dataset. In the case of the given example, the separation region in Fig.4.1 (a) is between time series *d* and *c*, and in Fig.4.1 (b) the separation region is between time series *d* and *b*. Therefore the clustering result while using the nearest seed is $\{e, a, b, d\}$ and $\{c\}$; while using the furthest time series seed produces clusters of $\{d, c\}$ an $\{b, e, a\}$. In this simple example, we are able to identify the intuitive and correct clustering result in one run using the furthest initialization. In contrast, the nearest initialization did not cluster time series d correctly. As this is only an example and therefore not sufficient enough to indicate that furthest initialization is always better than the nearest, we will perform more extensive experiments later on. With larger datasets, the situation could be more complicated, we cannot expect to get the final result with one split, and the clustering is typically further adjusted with DBA and iterated multiple times.

4.2 Cluster Update

As shown in Algorithm 4.1, Distance Density clustering is a definitive clustering method and is initialized by finding the time series that has the most number of either the furthest or nearest neighbors. Once the initial seed and clusters are obtained, the dataset is split, and two new seeds are selected by identifying the medoid of the DBA of that particular cluster. Then each time series are reassigned to the new seeds based on their DTW similarities.

With the initial seed, we obtain two clusters; we treat each cluster separately and repeat the sparse region split within each cluster. To avoid doubling the number of clusters with each iteration, we only cut at the stronger bend from the two similarity plots and identify one new cluster seed. The clusters would then be rebalanced based on the similarity between all the time series and the updated cluster seeds, and the algorithm would then go into the next iteration. This process is repeated until it has either reached a user set threshold, or no more clusters can be identified. Because DDC is an incremental process, the results can be interpreted as forming a divisive hierarchy, making the results more interpretable as well as not being constrained to the initial parameter settings, as is the case with k-means or k-medoids.

Although we use DBA, which is a global time series average, it is still sensitive to the computation template. For a completely deterministic clustering technique, we use the time series with the most nearest neighbors as the initial DBA template. The intuition behind this is based on the idea that time series averaged sequence is more likely to come from a denser region of the dataset. Similar to k-means or k-medoids, any meaningful help toward initialization is better than pure random. Not only will the DBA results be reproducible, but it could also potentially speed up DBA convergence.

Algorithm 4.1 Distance Density Clustering Algorithm

```
Input: $E = \{e_1, ..., e_n\}$ time series events to be clustered
Input: $C_{k-1} = \{c_1, ..., c_{k-1}\}$ set of cluster seeds
Input: number of seeds $k$
Input: $L_k$ is the cluster set of events based on the number of groups
 1: for do $l \in L_{k-1}$
 2:     $L_{k-1} \leftarrow \mathrm{Cluster}(C_{k-1})$
 3:     $ar[1, 2, ..., k-1] = \mathrm{DistSort}(L_{k-1})$
 4:     $value[i] \leftarrow max(arr[2] - ar[1], ..., ar[k-1] - ar[k-2])$
 5:     if $ar[n] - ar[n-1] == max(value[i])$ then
 6:         $location[i] = n$
 7:     end if
 8: end for
 9: if then $i \leftarrow max(value[1, ..., k-1])$
10:     $l(i_1, i_2) \leftarrow l(i), (c_{i_1}, c_{i_2}) \leftarrow c_i$
11: end if
12: return $L_n = \{1, 2, ..., i_1, i_2, ..., n\} \leftarrow C_k\{(c_1, c_2, ..., c_{i_1}, c_{i_2}, ..., c_n)\}$
13: for $e_i \in E$ do
14:     $(c'_1, c'_2, ..., c'_k) \leftarrow DBA(c_1, c_2, ..., c_{i_1}, c_{i_2}, ..., c_{k-1})$
15:     $\mathrm{UpdateClusterDBA}(C_k)$
16: end for
17: return $C'_k = \{c'_1, ..., c'_k\}$ as set of cluster seeds
18: return $L_n = \{l(e) \mid= 1, 2, ..., n\}$ set of cluster labels of $E$
```

As we have previously mentioned, the DDC method is divisive in structure; therefore, when more clusters are obtained, the performance would naturally increase. In the extreme case, when each event forms its own cluster, the clustering results would degenerate to a kNN (k=1) case. While this would increase the training accuracy dramatically, it is not useful in real-world applications to test every new piece of data to the entire existing dataset. The overwhelming amount of comparisons would also defy the purpose of clustering since users are unable to obtain a generalized understanding of the dataset. Therefore, in many cases, accuracy is not the goal of clustering.

4.3 DDC Performance

The performance of different initialization techniques are shown in Fig. 4.2 using (a) accuracy, (b) F score, and (c) rand index. The 50 datasets are labeled on the x-axis, and the y-axis is the measurement values. The boxed region shows the performance range for k-means with 100 runs of random initialization. The orange dots in each figure show the performance of furthest seed initialization, and the blue dots show the performance of the nearest seed initialization. Overall furthest seed initialization shows the best performance, while the nearest seed initialization usually falls somewhere within the random initialization performances. There are some patterns where certain datasets have higher performance regardless of the initialization method or the evaluation measure; it is possible that some datasets show more substantially inherent similarity and dissimilarity, and are merely easier to cluster.

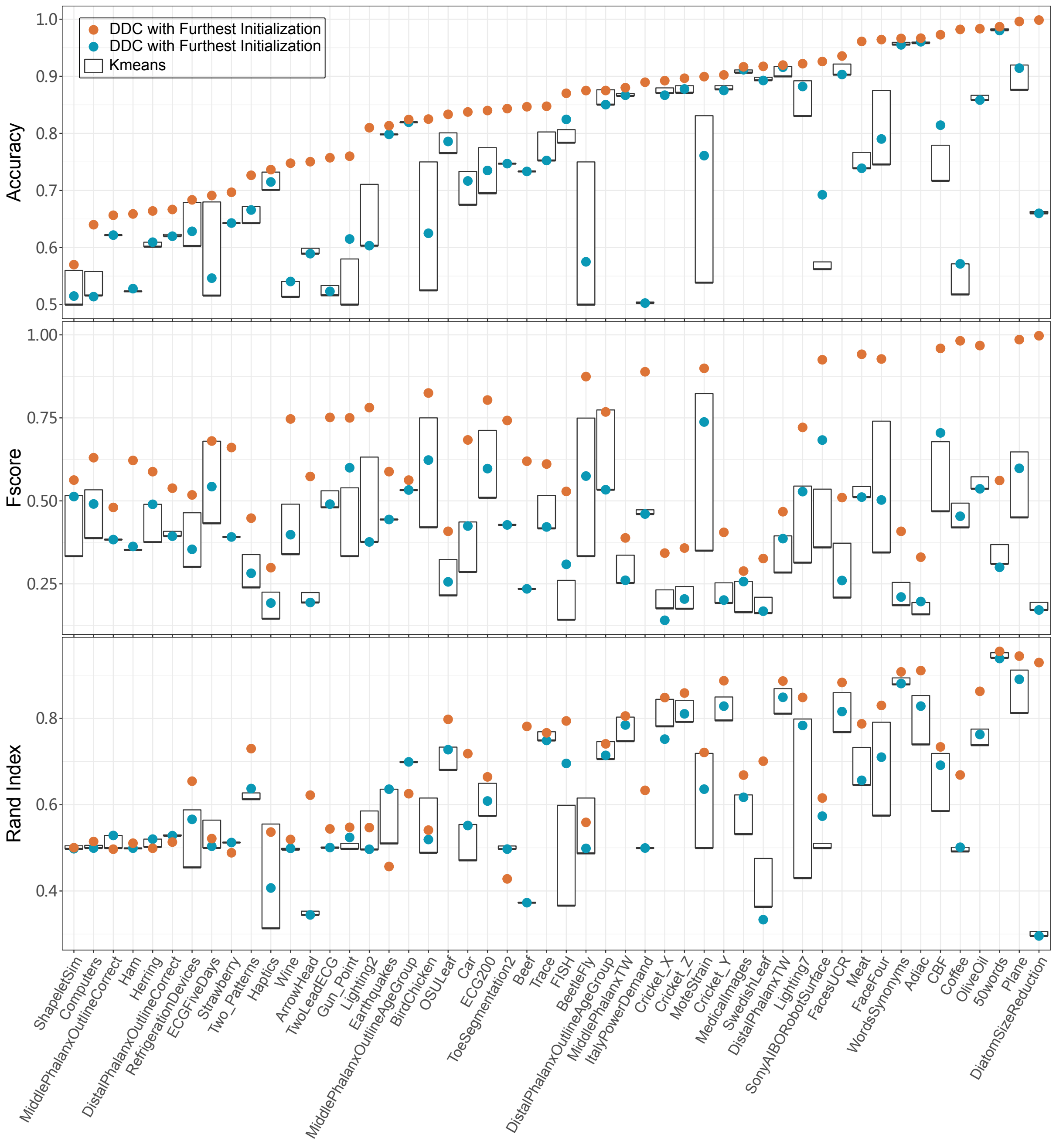


Figure 4.2: The Accuracy, F-score, and Rand Index of 50 datasets from the UCR repository. Using Distance Density clustering, the orange dot is the performance of furthest seed initialization; the blue dot is the performance of nearest seed initialization, and the box region is the performance range of k-means with random initialization.

5 CLUSTER QUALITY EVALUATION

As introduced in Chapter 2, most internal indices such as the Silhouette Coefficient, Dunn Index, Davies-Bouldin Index, etc. utilize some combination of inter-cluster distance and intra-cluster distance. This may become a potential problem with time series data. By nature, time series is high-dimensional, and therefore the distribution of time series clusters is challenging to visualize. The complex combination of inter-cluster distance and intra-cluster distance of high-dimensional data means the expressiveness of internal indices could be greatly reduced.

Traditionally, when internal indices are applied for discrete data, the utilized inter- and intra-distances in the Euclidean space conforms with the triangle inequality theorem. As is shown in Fig. 5.1, given a proper triangle with positive side lengths a, b, and c, without considering the case where the triangle area is zero, the sum of any two sides is greater than the remaining side. This can be extended to include the case where the triangle area is zero, in which case the sum of any two sides is greater than or equal to the remaining side. In other words, the distance between any two points in a two-dimensional plane is less than or equal to the summed distance from the two points to a third point.

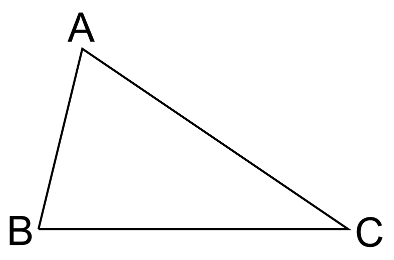


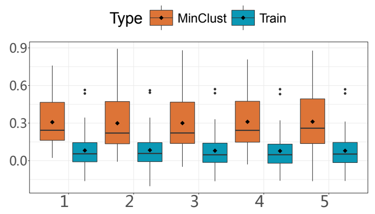
Figure 5.1: Triangle inequality theorem: given a proper triangle with side lengths a, b, and c that are all positive and excludes the degenerate case of zero area, the sum of any two sides is greater than the remaining side.

5.1 Data Analysis

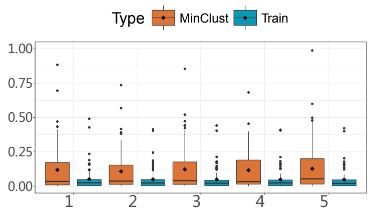
Many of the UCR archive datasets are real-world data, and therefore, the given labels are regarded as the ground truth. Based on this assumption, we first analyze the datasets. When handling real-world data, very often, the datasets are human-labeled. Meaning there is the possibility of mislabeling. Additionally, even with correct labeling, different datasets also have different innate quality. In many instances, we find that some datasets have very high clustering/classification accuracy regardless of the applied heuristic, while other datasets could have low accuracy despite the applied heuristic.

Although time series is a very different data format from discrete data, the idea of clustering is identical. Therefore, to analyze the effectiveness of cluster evaluation indices, we assume that with a suitable similarity measure, both internal and external indices should work for time series data cluster evaluation. For a naive comparison, we use the given ground truth labels as cluster assignment and compare that with the minimal clustering produced by DDC, which was introduced in Chapter 4. For the minimal clustering, we generate the same cluster of numbers as the number of class labels, meaning regardless of the dataset size, if there are three class labels, we generate three clusters. Unless we are processing an extremely ideal dataset, minimal clustering should have inferior performance; therefore, this serves as a baseline for comparison. Fig. 5.2 shows the Silhouette Coefficient, the Dunn Index, and the Davies-Bouldin Index of 71 datasets, the x-axis is the five groups from the five-fold cross-validation, and y-axis is the corresponding index value. Boxplots are labeled as MinClust and Train, which corresponds to the minimal cluster case and the class label assignment case. With all three indices, minimal clustering demonstrated better performance, which suggests the complex distribution of the tested time series datasets.

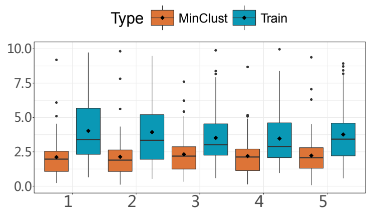
As demonstrated in Fig. 5.3, for any dataset there are three possible distributions; firstly, in the worst circumstance, each natural cluster is a mix of class events; the second case is



(a) Silhouette Coefficient

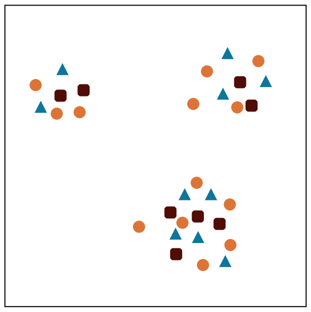


(b) Dunn Index

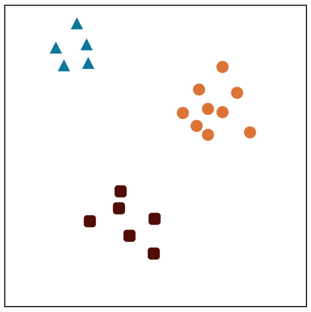


(c) Davies-Bouldin Index

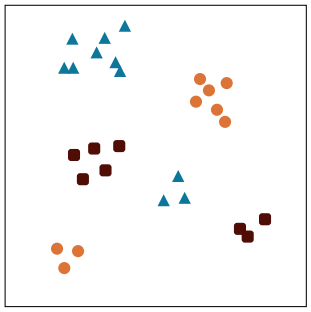
Figure 5.2: A comparison between minimal DDC and class label assignment, here minimal clustering refers to producing the same number of clusters as there are class labels, (a) shows the Silhouette Coefficient, (b) shows the Dunn Index, and (c) shows the Davies-Bouldin Index. For all three indices, minimal clustering showed better performance than the ground truth class label assignment.



(a) Case 1



(b) Case 2



(c) Case 3

Figure 5.3: Three types of distribution, (a) multiple classes belonging to the same natural cluster, (b) class label coincide with natural clusters, (c) multiple natural clusters forming the same class.

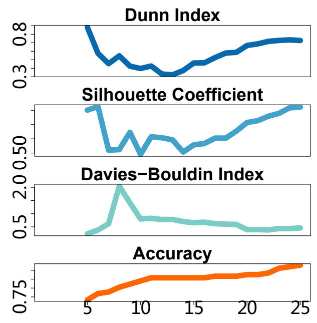
where the number of natural clusters coincides with the number of class labels; the last case is where multiple natural clusters exist for the same class label. In the case that there are multiple classes within one cluster, we should expect to see a worse performance from naive clustering than from original class label separation, due to the clustering algorithm can not discern any natural separations. From the poor performance of using class label assignment, we can deduce that each labeled class is likely made up of several clusters. This situation is not unique to, but it is rather harder to identify for time series data. The results also confirm our assumption that the distribution of time series data is complex. When the data is high-dimensional, the distribution is high-dimensional.

5.2 Clustering Validation with Internal and External Indices

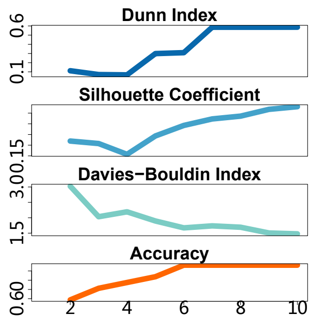
Clustering analysis serves the purpose of identifying groups of similar objects and patterns. Through clustering time series datasets, we aim to discover correlations and distributions. Internal indices were developed to help identify a good number of clusters when no labels are available. For internal cluster evaluation, we used five-fold cross-validation for each dataset. Because no labeled ground truth is considered for internal validation, only four-sections of each partition is used for clustering. Here we assess the performance of the internal indices with time series clusters.

As is shown in Fig. 5.4, the internal indices of clustering validation performs poorly for time series clusters, and it is difficult to identify a suitable cluster number for cluster termination. This phenomenon could be explained by the high-dimensionality of time series data. The internal indices that are shown here are commonly used indices for crisp discrete data clustering and are known to provide satisfactory results for well-separated clusters. However, the way in which internal indices computes the values is not compatible with the complex structure of time series data. When the data is high-dimensional, the distribution is high-dimensional, and considering that neither time series nor DTW has triangle inequality, the clustering distribution cannot be evaluated with measures that combine intra-cluster and inter-cluster distances.

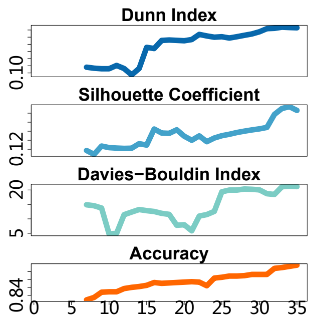
External evaluation involves *a priori* knowledge of the data. Meaning we will incorporate the ground truth label from the datasets to evaluate the formed clusters. The external measurement of accuracy is used here as a guideline to interpret the values of internal indices. Although the trend of accuracy increase with cluster numbers, it is not a linear process. A cluster number increase may result in worse distribution of data clusters, which we can observe as a decrease in accuracy. We can observe from Fig. 5.4 that internal index values demonstrate much more dramatic changes than that of the accuracy. This could be due to the nature of two index categories; the internal index represents the



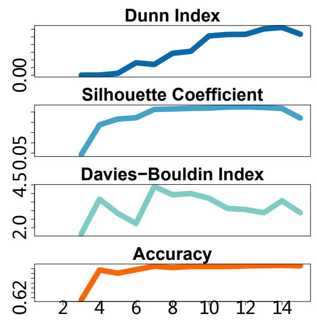
(a) Beef



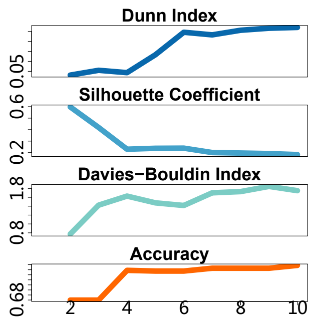
(b) BeetleFly



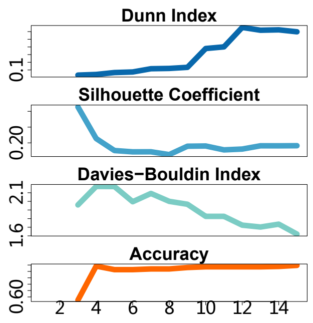
(c) FISH



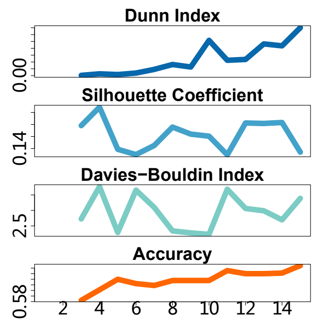
(d) LgKitchenApp



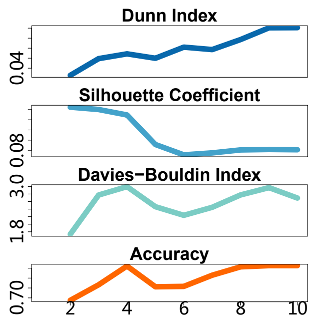
(e) ProxiPhalOutCorr



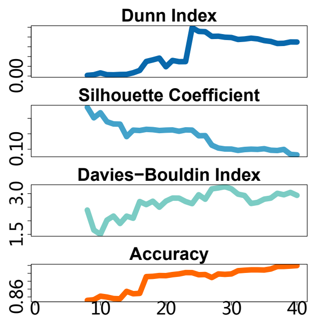
(f) Referigeration



(g) ScreenType



(h) SonyAIBORobotSurface



(i) uWaveGestureLibraryZ

Figure 5.4: Evaluation of clustering with internal indices values and the corresponding accuracy.

structure of clusters more closely, whereas the external index relies on given class labels. When natural clusters do not strictly correspond to given labels, the response of external indexes improvement could be postponed. In other words, sometimes, the qualitative change of cluster structure needs to accumulate before leading to the quantitative change for accuracy improvement.

The goal of clustering is to find natural groupings of high homogeneity, with many datasets, especially real-world datasets, it is very rare to have neatly separated clusters. In

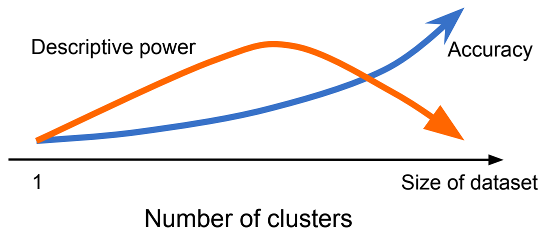


Figure 5.5: Accuracy and descriptive power in relation to the number of clusters.

application, there is a trade-off between the descriptive power of clusters and achieving high accuracy. This relationship is approximated in Fig. 5.5. When all objects belong to the same cluster, there is no descriptive meaning and very low accuracy. Generally, the more clusters there are, the higher the training accuracy. In the extreme case, when each object is its own cluster, the training accuracy is 100%. However, clustering ceases to be a learning procedure and becomes a memorization process; in other words, it is no longer descriptive and becomes 1NN.

5.3 Cluster Variance Evaluation

Most existing internal indices work under the assumption that all points exist in a two-dimensional plane. However, time series, along with the DTW algorithm, does not conform with the triangle inequality. As such, it would seem irrational to expect evaluation measures adapted to triangle inequality theorem to work for high dimensional time series data clusters. Therefore, we propose the use of a variance-based evaluation method for time series cluster evaluation. Despite its unpredictable distribution, we stay close to the basic definition of clusters, as time series clusters would still comply with the basic characteristics of clustering, which is high association with intra-clusters and low association with inter-clusters.

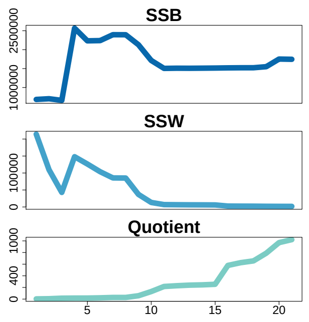
When using variance as a measurement for clustering we utilize between cluster variance SS_b (Eq. 5.1) and within cluster variance SS_w (Eq. 5.2), with the cluster mean computed with DBA. In order to emphasize the separation of inter-cluster and intra-cluster measurements, the total variance $SS_t = SS_w + SS_b$ is not utilized, as SS_t suggests the establishment of the triangle inequality. We could evaluate the cluster quality with two measurements; however, for simplicity, we use the quotient Q to combine the two variances. Shown in Eq. 5.3, Q combines the two variances SS_b and SS_w only after the two values are computed separately, and it is not a combination of inter-cluster and intra-cluster similarities, but rather a simple measurement figure. Based on the definition of clustering, a large value of SS_b and a small value of SS_w is preferred, meaning a larger Q indicates clearer distinction among clusters.

$$SS_b = \sum_{i=1}^{m} \sum_{j=1}^{n_i} (\overline{X}_i - \overline{X})^2 \tag{5.1}$$

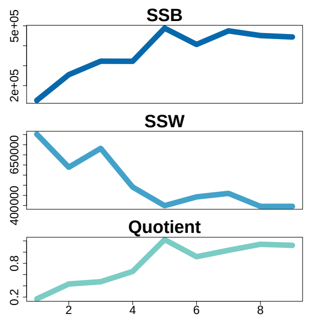
$$SS_w = \sum_{i=1}^{m} \sum_{j=1}^{n_i} (X_{ij} - \overline{X}_i)^2 \tag{5.2}$$

$$Q = SS_b / SS_w \tag{5.3}$$

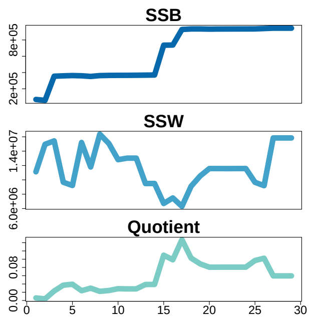
Fig. 5.6 shows the variance values for 9 datasets. As a general trend, SS_b increases, and SS_w decreases as the number of clusters increases. Ideally, the maximum SS_b and minimum SS_w coincide at the same cluster assignment; in less ideal situations, we look for a balance between the two values. We use Q to find the point where the ratio of a large SS_b and a small SS_w becomes significant, the largest value or the knee point. Here the compactness and separation are expressed with within cluster variance and between



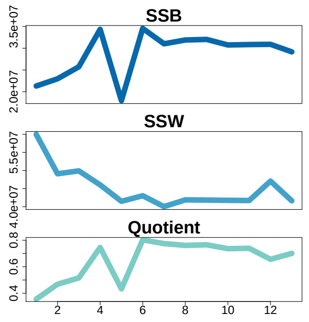
(a) Beef



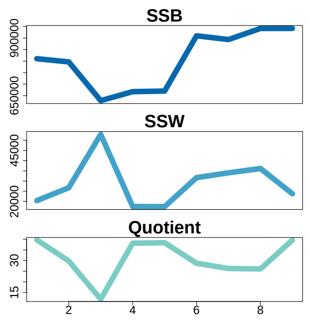
(b) BeetleFly



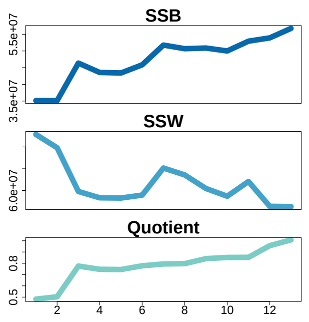
(c) FISH



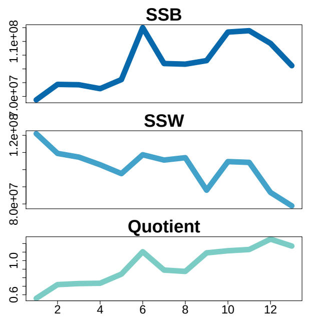
(d) LgKitchenApp



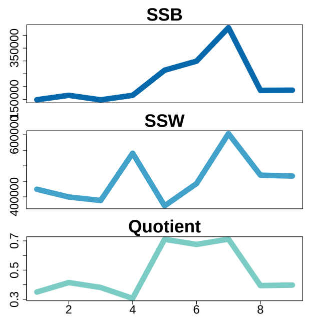
(e) ProxiPhalOutCorr



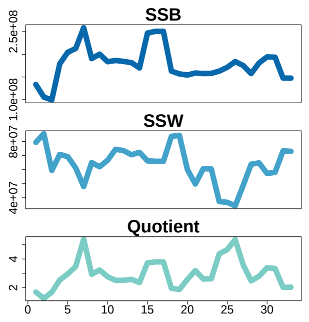
(f) Referigeration



(g) ScreenType



(h) SonyAIBORobotSurface



(i) uWaveGestureLibraryZ

Figure 5.6: Evaluation of clustering with variance values.

cluster variance. When they are independently considered before merging the final value, we can observe a separate indication of cluster quality.

For some datasets, there is one significant maximum Q that indicates a suitable place to stop clustering, whereas for other datasets, there are multiple choices. Considering the cost of clustering as well as the possibility of losing the descriptive power of clusters, when having similar performance, a smaller number of clusters is more often preferred.

6 EXPERIMENTS AND APPLICATIONS

In this chapter, we present our applicational experiments. First, a comparison study between AGNES and OPTICS is conducted in Section 6.1, where the hierarchical structure correlation is explored. Then we discuss the extension and applications of time series clustering. With an effective distance measure and an adequate averaging technique, we can achieve fairly good clustering on univariate time series data. However, in application, when we need to generate predictions, it is usually not enough to perform univariate clustering on high dimensional data. Here high dimension refers to multivariate datasets. Decision trees are often used with high dimensional data for decision processes. However, traditional decision trees are usually implemented with categorical or numeric data, and not time series data. Section 6.2 briefly discuss the decision tree heuristic. A cluster based decision tree for ICME and solar flare data are presented in Sections 6.3 and 6.4. Finally, Section 6.5 shows how we use normalization to identify time series trends, which can be used to create event class profiles and has the potential for real-time prediction.

6.1 Hierarchical Structure Comparisons

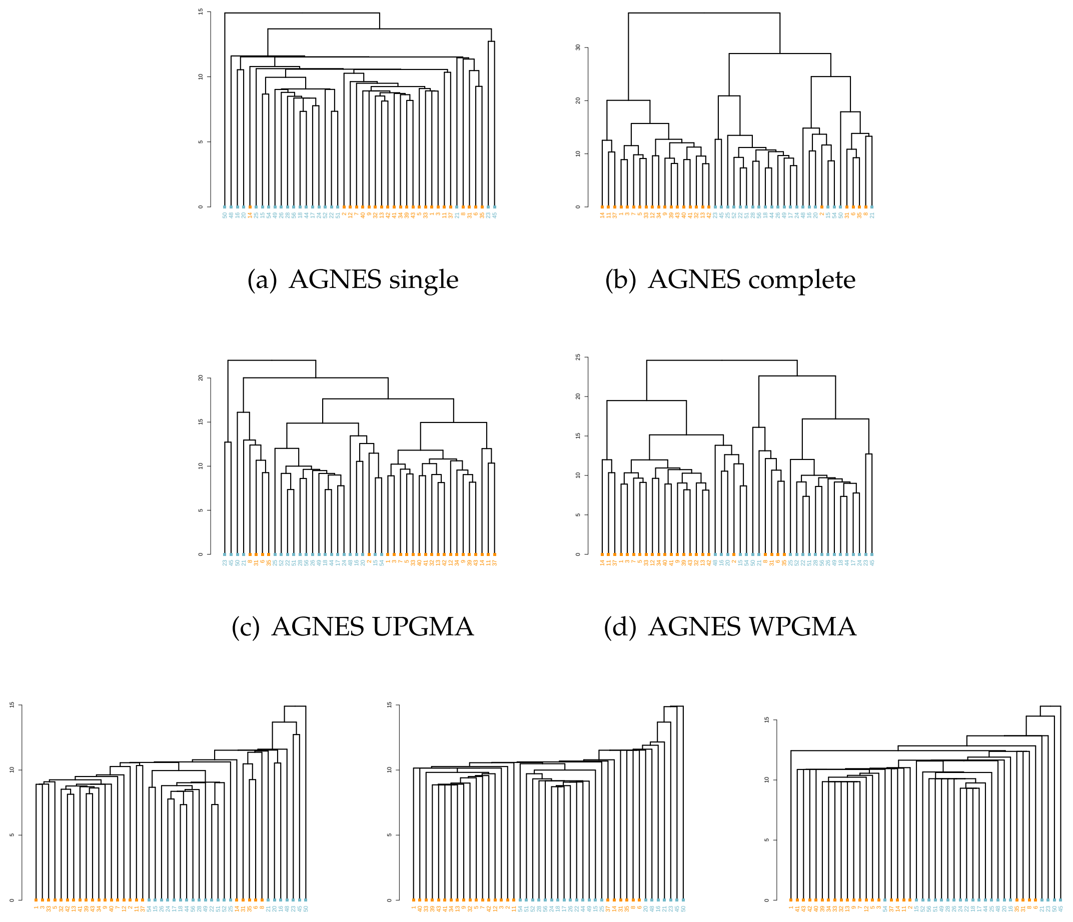
For hierarchical structure analysis, we use datasets from the UCR repository [21]. Because cluster analysis requires balanced datasets, we used the re-sampled datasets. Due to the computation costs as well as the difficulty to sensibly visualize large datasets, our experiments are limited to small to medium-sized datasets, i.e., hierarchical clustering is not suited for larger datasets.

Our goal is to comprehend how different hierarchical structure performs with time series data. Because of the structural similarity, we compare AGNES with single, complete, UPGMA and WPGMA linkage, and OPTICS with $\epsilon = \mathtt{Inf}$ and $\mathtt{minPts} = \{1, 5, 10\}$. Specifically, we evaluate the dendrogram quality when compared to the original dataset similarities. Then the dendrograms from different implementations of AGNES and OPTICS are compared for similarity. Finally, we use class labels to evaluate the quality of cluster partitions, as well as the robustness of different clustering algorithms.

6.1.1 Internal Evaluation

First and foremost, we look at the dendrograms produced by each implementation of AGNES and OPTICS. Although we cannot accurately compare dendrograms, they do provide a visually intuitive understanding. Due to the space limitation and the massive quantities of dendrograms, we only use one dataset as an example, dataset "Coffee" from UCR. Fig. 6.1 show AGNES with single, complete, UPGMA, and WPGMA linkage, as well as OPTICS with $\epsilon = \mathtt{Inf}$ and $\mathtt{minPts} = \{1, 5, 10\}$. Visually, we can observe certain similarities between the OPTICS dendrograms. With slight differences, the elements in the OPTICS dendrograms are more concentrated, with more noticeable outliers and less distinct dissimilarity between larger clusters. Also, we can observe the similarity between AGNES complete, UPGMA, and WPGMA, in the sense that there are more distinct and balanced larger clusters. The AGNES single dendrogram is visually more similar to the OPTICS dendrograms, upon further inspection, the dendrogram AGNES single is identical to OPTICS with $\mathtt{minPts} = 1$, just with a different ordering of presentation. We observed the identical dendrograms between AGNES single and OPTICS $\mathtt{minPts} = 1$ for all the datasets we tested and will discuss the cause for this phenomenon later on.

Table 6.1 contains the cophenetic coefficient of the corresponding dendrogram compared to the similarity matrix of 25 datasets. This is a simple evaluation of the preservation


(a) AGNES single (b) AGNES complete

(c) AGNES UPGMA (d) AGNES WPGMA

(e) OPTICS minPts=1 (f) OPTICS minPts=5 (g) OPTICS minPts=10

Figure 6.1: Dendrogram of four implementations of AGNES, and three implementations of OPTICS.

of similarity in the respective dendrograms. From the CP evaluation, we can see overall better similarity preservation with UPGMA, WPGMA, and complete linkage. Of course, depending on the specific dataset, we do observe exceptions. The similarity preservation of different OPTICS setting is comparable with no apparent leader. We can observe that OPTICS with $\mathtt{minPts} = 1$ has the same CP values as AGNES with single linkage.

Furthermore, we use both the CP coefficient and the Goodman Kruskal's gamma to quantitatively compare the dendrograms from different clustering methods for dataset "Coffee". Shown in Fig. 6.2 and Fig. 6.3 are the CP matrix and the Goodman Kruskal's

Table 6.1: Cophenetics Values for Variations of OPTICS and AGNES.

DataSet	OPTICS			AGNES			
	minPts = 1	minPts = 5	minPts = 10	Single	Complete	UPGMA	WPGMA
ArrowHead	0.7777	0.7849	0.8097	0.7777	0.2092	0.8538	0.6935
Beef	0.8664	0.8511	0.8535	0.8664	0.8835	0.8951	0.8827
BeetleFly	0.4126	0.4483	0.2517	0.4126	0.4147	0.6021	0.5784
BirdChicken	0.6239	0.5774	0.4883	0.6239	0.5682	0.7105	0.6231
Car	0.2751	0.3765	0.4039	0.2751	0.4170	0.6841	0.6280
Coffee	0.7008	0.6523	0.6398	0.7008	0.6311	0.7360	0.6519
DiatomSizeReduction	0.9486	0.9489	0.9492	0.9486	0.9573	0.9767	0.9765
Earthquakes	0.4120	0.4157	0.4204	0.4120	0.6832	0.7212	0.6847
ECG200	0.6185	0.6535	0.6657	0.6185	0.7038	0.8145	0.6685
FaceFour	0.7603	0.7682	0.7552	0.7603	0.6091	0.8321	0.8110
FISH	0.0227	0.0222	0.0223	0.0227	0.4956	0.4980	0.4751
Gun_Point	0.7917	0.8112	0.8507	0.7917	0.8333	0.8677	0.8130
Ham	0.6120	0.6125	0.6145	0.6120	0.4196	0.7128	0.6403
Herring	0.0624	0.0414	0.0743	0.0624	0.5383	0.5649	0.5646
Lighting2	0.3157	0.3643	0.3846	0.3157	0.4755	0.5621	0.5466
Lighting7	0.3737	0.3664	0.4406	0.3737	0.6302	0.6903	0.6815
Meat	0.8368	0.8157	0.8209	0.8368	0.8270	0.8863	0.8785
OliveOil	0.8983	0.8925	0.8669	0.8983	0.8836	0.9325	0.9116
OSULeaf	0.2668	0.2882	0.2860	0.2668	0.3994	0.5093	0.3656
Plane	0.7666	0.7669	0.7673	0.7666	0.4457	0.8972	0.6710
ShapeletSim	0.1546	0.1456	0.1818	0.1546	0.2806	0.3944	0.3350
ToeSegmentation1	0.4023	0.4060	0.4213	0.4023	0.2867	0.5782	0.3826
ToeSegmentation2	0.3420	0.3643	0.3732	0.3420	0.3464	0.4968	0.4849
Trace	0.7918	0.7913	0.7897	0.7918	0.8063	0.8890	0.8846
Wine	0.7234	0.7015	0.7021	0.7234	0.7313	0.8374	0.7945

gamma matrix illustrating the similarity between different hierarchical clustering methods. Again, we can observe that OPTICS with $minPts = 1$ and AGNES with single linkage are identical. Also, AGNES with complete and WPGMA linkage is the least similar to other dendrograms. Note that this comparison is between the similarity of different clustering dendrograms, and is not a measurement of the quality of the dendrograms.

From the internal evaluation of dendrograms for different implementations of AGNES and OPTICS, we observed several characteristics in the context of the time series data that we experimented with. As the $minPts$ value in OPTICS approach the value of 1, OPTICS becomes more similar to AGNES with single linkage. One of the characteristics of AGNES single link is the chaining effect. When the most dense elements within a dataset are

$$\begin{array}{r|ccccccc} & \mathrm{O1} & \mathrm{O5} & \mathrm{O10} & S & C & U & W \\ \hline \mathrm{OPT1} & 1.00 & & & & & & \\ \mathrm{OPT5} & 0.96 & 1.00 & & & & & \\ \mathrm{OPT10} & 0.88 & 0.93 & 1.00 & & & & \\ Single & 1.00 & 0.96 & 0.88 & 1.00 & & & \\ Complete & 0.41 & 0.31 & 0.32 & 0.41 & 1.00 & & \\ UPGMA & 0.77 & 0.71 & 0.72 & 0.77 & 0.76 & 1.00 & \\ WPGMA & 0.50 & 0.41 & 0.45 & 0.50 & 0.77 & 0.75 & 1.00 \end{array}$$

Figure 6.2: Cophenetic coefficient matrix showing the similarity between dendrograms from different hierarchical clustering methods for dataset "Coffee".

$$\begin{array}{r|ccccccc} & \mathrm{O1} & \mathrm{O5} & \mathrm{O10} & S & C & U & W \\ \hline \mathrm{OPT1} & 1.00 & & & & & & \\ \mathrm{OPT5} & 0.96 & 1.00 & & & & & \\ \mathrm{OPT10} & 0.80 & 0.83 & 1.00 & & & & \\ Single & 1.00 & 0.96 & 0.80 & 1.00 & & & \\ Complete & 0.36 & 0.31 & 0.36 & 0.36 & 1.00 & & \\ UPGMA & 0.75 & 0.74 & 0.81 & 0.75 & 0.58 & 1.00 & \\ WPGMA & 0.42 & 0.39 & 0.47 & 0.42 & 0.69 & 0.58 & 1.00 \end{array}$$

Figure 6.3: Goodman and Kruskal's gamma matrix showing the similarity between dendrograms from different hierarchical clustering methods for dataset "Coffee".

joint first in OPTICS, and $\mathtt{minPts}$ value is low, the cluster growth is similar to joining clusters based on the nearest elements in those clusters. This characteristic contributes to OPTICS and AGNES single linkage algorithms to identify arbitrarily shaped clusters. As the $\mathtt{minPts}$ threshold is raised, the reachability structure is altered. This can also be observed as larger $\mathtt{minPts}$ values produce smoother reachability plots.

Overall, AGNES UPGMA, WPGMA, and complete linkage have the best overall ability to maintain dendrogram similarity in comparison to the original similarity matrix. For dendrogram similarity, we can observe OPTICS $\mathtt{minPts} = 1$ being identical to AGNES single linkage. As the $\mathtt{minPts}$ value is increased, the similarity to AGNES single linkage is decreased. AGNES UPGMA linkage has some similarities to all the other tested methods, while the complete and WPGMA linkage produced the most distinct dendrograms.

6.1.2 External Evaluation

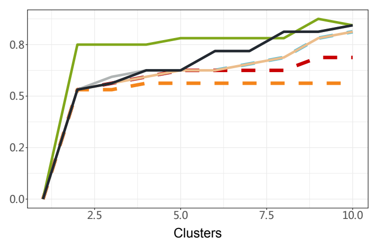
So far, we evaluated the performance of different hierarchical clustering methods based solely on dendrograms. Another way to evaluate cluster quality is external evaluation, where *a priori* knowledge is considered. In this section, we utilize the given class labels to compare the cluster quality from different cluster heuristics. As space is limited, three datasets' visualizations are shown as a representation, as there are definite trends among different datasets, and the visualization can benefit the understanding of the performance of different clustering methods.

For accuracy evaluation, we first generate a dendrogram from the training portion of the data. Then the dendrogram tree is cut at different heights to generate the respective number of clusters. For each different cluster partition, a time series average DBA is generated to represent each training cluster. In order to reasonably evaluate the clustering performance, the testing portion of data is invisible to the training process. Testing data is compared against each of the cluster averages and assigned the label of the cluster it is most similar to. The compliance between the assigned label and the actual label determines the testing performance.

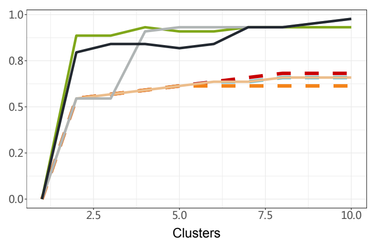
During the training cluster generating process, it is observed that not all numbers of clusters exist. For example, we can obtain 4 clusters and 6 clusters from a certain dendrogram, but not 5 clusters. This can be explained by the incredibly dense cluster formations generated by OPTICS dendrograms. As suggested in the previous dendrogram visualization results, some branches have multiple elements joint at the same height, meaning certain numbers of cluster cuts simply do not exist. In which case, the accuracy of the previous number of clusters is used for accuracy curve visualization.

We generate accuracy curves for both the training process and the testing process. Fig. 6.4 shows the accuracy curves for three datasets. On a local scale, the accuracy curves do not necessarily grow monotonously. Although the global scale performance improves

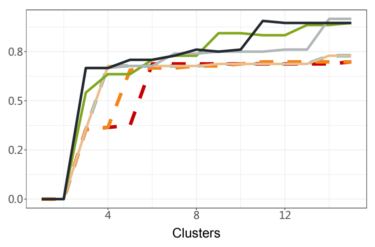
Methods
AGNES_Single
AGNES_Complete
AGNES_UPGMA
AGNES_WPGMA
OPTICS_minPts1
OPTICS_minPts5
OPTICS_minPts10



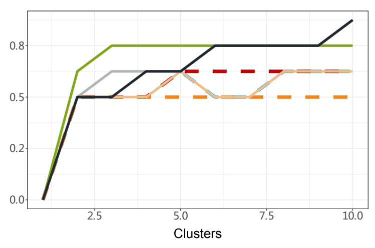
(a) Training curve for "Bird-Chicken".



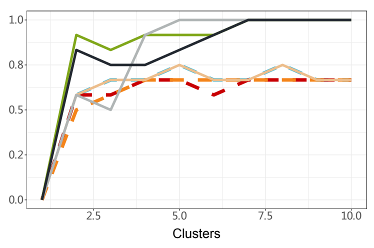
(b) Training curve for "Coffee".



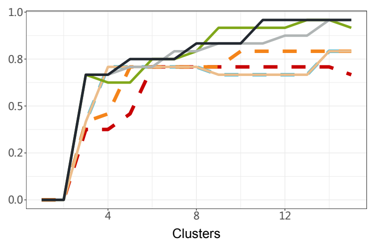
(c) Training curve for "Meat".



(d) Testing curve for "Bird-Chicken".



(e) Testing curve for "Coffee".



(f) Testing curve for "Meat".

Figure 6.4: The training and testing accuracy curves for datasets: "BirdChicken", "Coffee", and "Meat", for the different number of clusters.

with the increase of cluster numbers, there are situations where an increase in cluster number does not guarantee an increase in accuracy. The small local decrease in accuracy is mostly created when a previous minority becomes the current majority only slightly at that particular partition and causes the current minority or testing elements to be wrongfully labeled. We can observe that AGNES with complete, UPGMA, and WPGMA linkage have an overall better performance; and that the performance of OPTICS and AGNES single linkage is relatively weaker. Consistent with previous results, the training and testing accuracy curve of OPTICS with $\mathtt{minPts} = 1$ is identical to and overlaps with, the AGNES single linkage accuracy curves. The performance of testing remains comparable to the training portion, meaning hierarchical clustering of time series is, for the most part, robust in performance.

We can observe the similarity between dendrograms of under-performing clustering methods, namely AGNES single link, and OPTICS. One explanation for the poor performance from density based hierarchical clustering could be attributed to the underlying way in which OPTICS form clusters. Both OPTICS and DBSCAN form clusters through reachability, which can be visualized as a linked chain of data elements. When it comes to time series, a linked chain of similarity may not be a suitable representation, as there is the possibility of varying shapes linked together through transitive similarity. In which case, both ends of the linked time series chain may actually be distinct from each other. The overall best performing clustering methods, in the sense of accuracy, are AGNES complete, UPGMA, and WPGMA linkage. All of which produce relatively distinct dendrograms in terms of CP and Goodman Kruskal's gamma coefficients.

6.2 Decision Trees

Decision trees are tree-like structures, which organize a decision process used for data predictions. Objects are partitioned into groups where instances are similar. Typically, a decision tree contains a root node, several internal nodes, and several leaf nodes. The leaf nodes contain the decision results, and all other nodes contain conditional tests on attribute values that split the instances among its child nodes. The path from the root node to each leaf node is a sequence of tests (e.g., decisions) that determines where each instance belongs. This is a divide-and-conquer process that usually ends when the majority of the instances in each leaf node belongs to the same class.

In order to build an efficient decision tree, it is preferable for the instances at each node to have the highest concentration of a certain group of data. This can be computed with information entropy, which is also informally known as impurity. A smaller entropy value suggests purer groups (e.g., subsets where the majority of events belong to a single class), and a larger entropy value suggests more mixed groups.

However, entropy and information gain are mostly applied to discrete attributes and not on sequential values. A common way around this problem is to translate sequential data to the discrete case. However, instead of using statistical summarizations of event parameters such as average and standard deviation, we believe clustering results contain more information about the original time series data than a discrete summarization. The significance of entropy is to differentiate the purity of a parameter, which could also be represented by the clustering result. The advantage of this approach is that the meaningful cluster results can be inherited and aggregated with a decision tree.

Although commonly used in data mining, decision trees are not well adapted for sequential data. Here, we implement a multivariate time series decision tree by aggregating univariate clustering results. The clustering results can be used to mimic information entropy and is also much more relevant to pass on to a decision tree than the average value of each time series, which does little to differentiate each sequence. In other words, a decision tree that uses clustering results is not a value-based decision tree, but rather a shape-based decision tree. In this section, we show two cases where a decision tree is used to aggregate univariate clustering results to form multivariate decisions; one is with ICME data [78], and the second with solar flare data [79]. For visualization purposes, the decision node splits are shown with DBAs.

6.3 ICME Decision Trees

In order to build the decision trees on multivariate time series data, we mimic entropy with univariate time series clustering results. Using the significant parameters and their corresponding knee points, the clustering results are summarized into a categorical table. We are able to train decision trees using CART [80] for both AGNES and DDC.

To avoid performance differences by chance, we applied 10-fold cross-validation with stratified sampling in our experiments. From our previous works [76,81], we found the average link measure for AGNES and the furthest initialization technique for DDC are the better choices for the two clustering methods. Therefore only results based on these specifications will be presented. Fig. 6.5 and Fig. 6.7 show one of the 10-folds of the accuracy curves of 16 parameters when using DDC and AGNES. The accuracy refers to the matches between the clustering assignment to the original labels. For each accuracy curve, the x-axis is the number of cluster numbers, the y-axis is the accuracy, and the title includes the parameter name and the corresponding area under curve (AUC). Here we refer to the area under the accuracy curve as AUC for simplicity and not the common usage of referring to the area under the ROC curve.

The hierarchical clustering method is agglomerative and initializes with one event in each cluster because there are 181 events; the accuracy would be 100% when the number of subgroups is 181. The Distance Density approach is a divisive one, and each new subgroup is identified by finding a new cluster seed. Therefore once a subgroup has only two events left, or when there are three events left in a subgroup, and two of the events are equal in distance to the clustering seed, the divisive clustering has no reason to produce micro-clusters. This means the distance-density approach cannot reach 181 subgroups, and therefore is not guaranteed to end with accuracy being 100%. Additionally, for real-world applications, there is no reason to cluster to, or near 181 clusters. Therefore we only show the accuracy results up to 100 clusters.

6.3.1 DDC Decision Tree

The accuracy curves of different parameters behave differently, as is shown in Fig. 6.5. Parameters with a larger AUC suggest it is a better descriptor to separate magnetic cloud and non magnetic cloud events. Since we clustered the events to 100 clusters and the

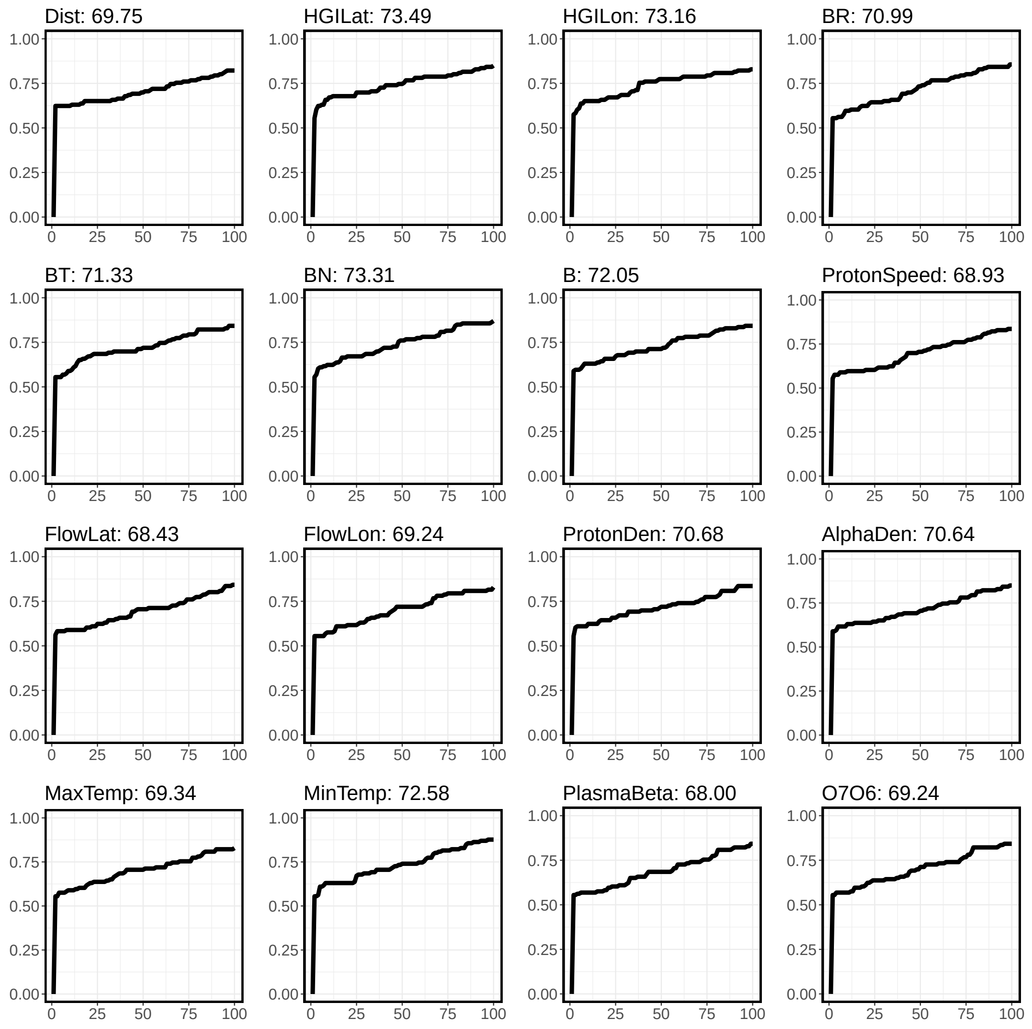


Figure 6.5: Distance Density accuracy curve for 16 parameters with AUC labeled.

highest accuracy value is 1, the highest possible AUC is 100. While we could feed all the parameters to a decision tree, this is generally avoided as it could result in complex trees that could lead to overfitting. Therefore, the most often occurring top five parameters with the largest AUC from each of the 10-fold splits are selected for decision tree training: *Bn*, *Bt*, *Alpha Density*, *B*, and *Br*.

The knee points from the accuracy curves are the points showing the highest accuracy increase by increasing one cluster. Only knee points with a cluster number less than 30 are considered because micro-clustering could lead to overfitting, and the most significant accuracy increase typically happens with fewer clusters. Then the selected parameters with the number of clusters based on its corresponding knee points form a categorical table shown in Table. 6.2, which is used to generate a decision tree.

Table 6.2: DDC categorical result table for building decision trees.

Parameter	Bn	Bt	Alpha Density	B	Br	Label
Event 1	Bn_4	Bt_3	AD_4	B_4	Br_1	MC
Event 2	Bn_3	Bt_2	AD_4	B_3	Br_6	MC
Event 3	Bn_2	Bt_3	AD_2	B_4	Br_6	MC
Event 4	Bn_2	Bt_3	AD_2	B_4	Br_4	MC
Event 5	Bn_1	Bt_3	AD_2	B_1	Br_1	NMC
...	...	...	...	...	...	...
Event 180	Bn_1	Bt_4	AD_3	B_1	Br_1	NMC
Event 181	Bn_2	Bt_1	AD_1	B_2	Br_2	MC

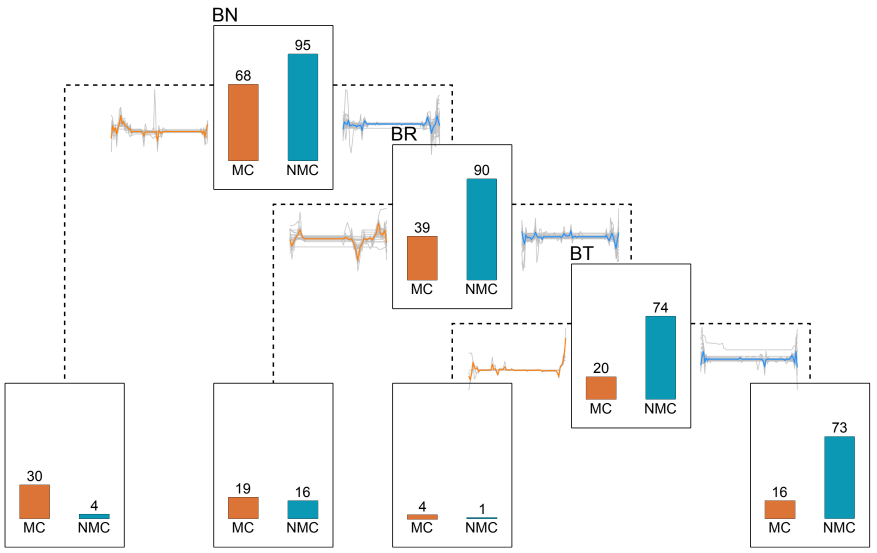


Figure 6.6: Shape-based multivariate time series decision tree for DDC.

Using the same dataset fold shown in Fig. 6.5. The DDC decision tree is shown in Fig. 6.6. Each non-leaf node is labeled with the parameter used for the split, the number of events in that node, and the ratio of magnetic cloud and non-magnetic cloud events at that node. Each leaf node has the number of magnetic cloud and non-magnetic cloud events distributed to it and labeled by the majority event class labels. Each branch in the decision tree is depicted with the DBA on the portion of events passed down to its child node, shown with the parent node parameter. The DBA is accompanied by the time series event sequences that are passed on the child node. The time series are matched to the group DBA using DTW to be stretched or compressed to a uniform length, which can help us look for time series patterns of significant peaks and valleys. The DBAs of magnetic clouds are colored orange, non-magnetic clouds are colored blue, and the stretched time series from that cluster is gray. We can see the DBAs display characteristic differences from branch to branch. This particular DDC decision tree has the training accuracy of 77.30%, and testing accuracy of 88.89%.

6.3.2 AGNES Decision Tree

For comparison, we repeated the process and generated decision trees for AGNES. Using the same fold of data for DDC, the parameter accuracy curves are shown in Fig. 6.7, and the corresponding AGNES decision tree is shown in Fig. 6.8. For the AGNES decision tree, the training accuracy is 80.37% and testing accuracy is 61.11%. This is a classic case showing the occurrence of overtraining, where a high training accuracy does not guarantee a comparable testing accuracy.

Overall, the DBAs depicting the non-magnetic cloud branches are more flat and straight compared to the magnetic cloud branches. More specifically, the time series in the root node split from the AGNES tree (Fig. 6.8) shows more difference to its DBA than the root node split in the DDC tree (Fig. 6.6), suggesting it is less effective than the DDC tree. Although appearing purer, three out of the five leaf nodes (the middle three) in the AGNES tree contain very few events, which implies overfitting, and also making it unreliable for testing or prediction usage.

In application, when new data comes in, only the corresponding parameters need to be matched to the DBAs on the decision tree branches. This means that prediction would be very efficient, since only a few of the parameters need to be checked, and computing the match between two sequences is less time-consuming. This can be very useful for future automatic magnetic cloud detection. For example, when the initial pattern or a significant

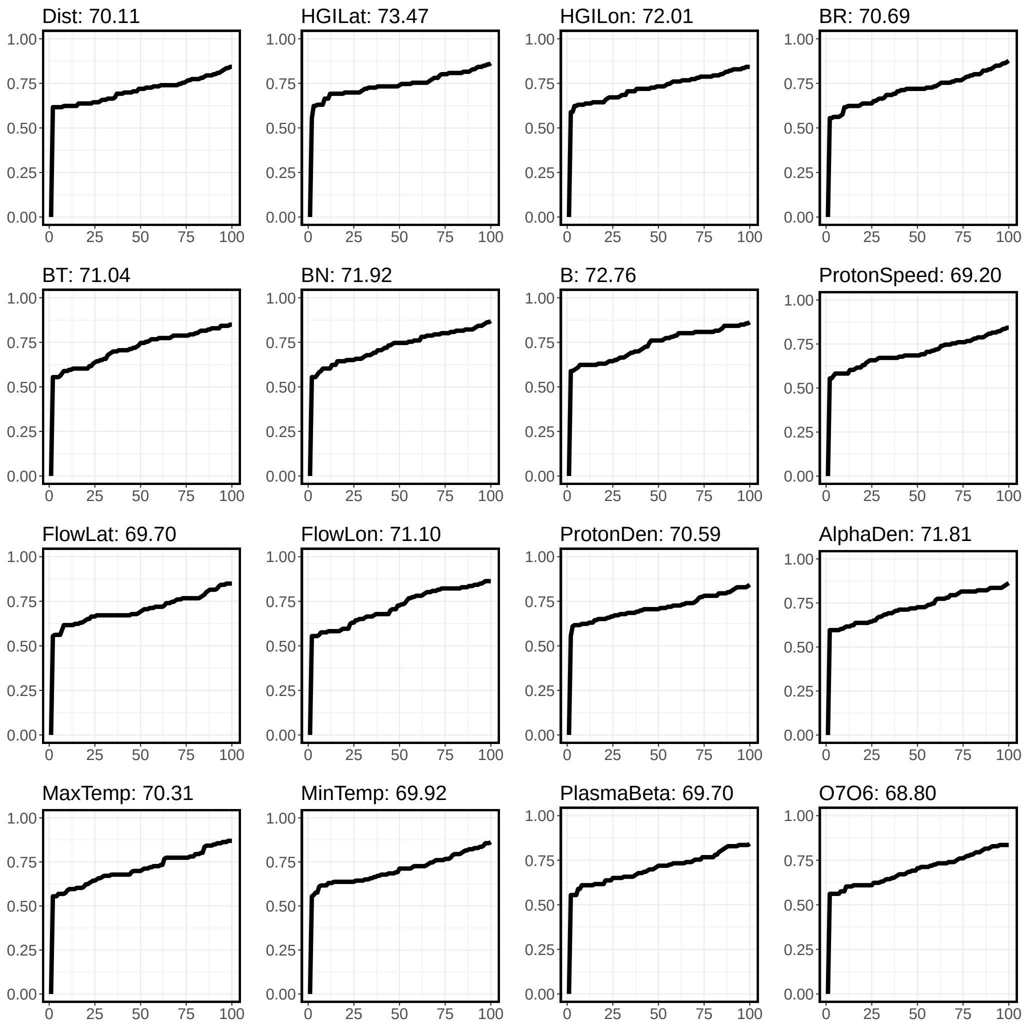


Figure 6.7: AGNES accuracy curve for 16 parameters with AUC labeled.

feature is matched between the incoming time series and the average time series of the parameters within a decision tree, the event could be labeled as a magnetic cloud event. With little to no human involvement, this automated magnetic cloud identification process could improve efficiency and reduce human-errors.

For comparison purposes, we computed the average value of each parameter of each event, and without clustering, generated a decision tree based on the numerical average values. In Table 6.3, we show the averaged training and testing accuracy of the 10-fold cross-validation process on the numeric decision trees, the AGNES decision trees, and the DDC decision trees. In Fig. 6.9, we provide a set of boxplots showing all of the training

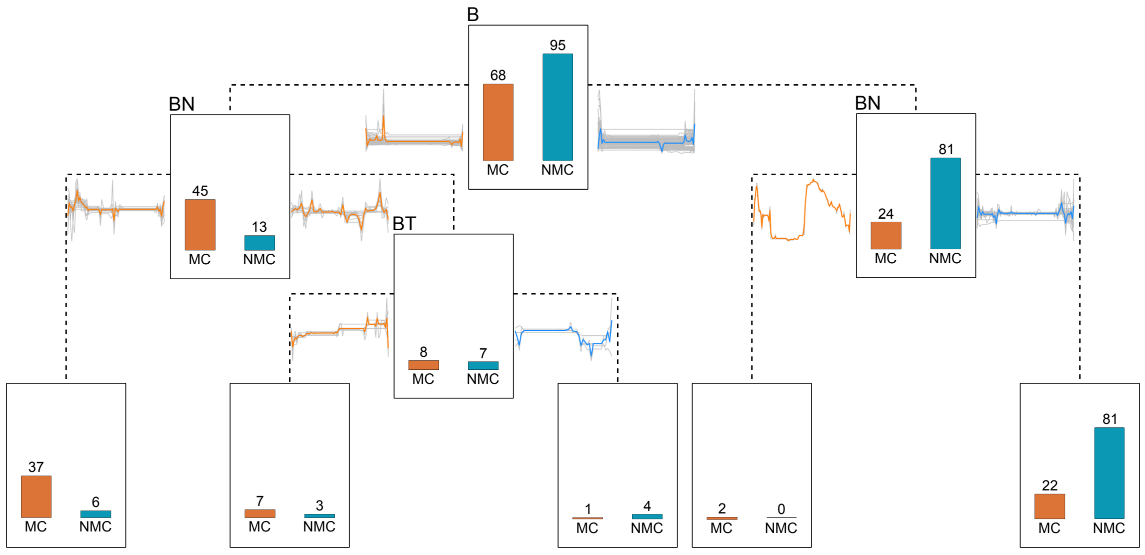


Figure 6.8: Shape-based multivariate time series decision tree for AGNES.

and testing accuracies for the numeric representation, the clustering using AGNES, and the clustering using DDC.

Table 6.3: Average Decision Tree Accuracy

Case	Training	Testing
Numeric Representation (No Clustering)	80.10%	71.87%
Agglomerative Hierarchical clustering	76.86%	73.98%
Distance Density Clustering	75.94%	75.12%

While the numeric data generated decision tree has the highest training accuracy, it has the worst testing accuracy in terms of average and interquartile range (IQR) variance. The AGNES decision tree is more stable than the discrete dataset tree, but also show signs of overtraining when the training and testing accuracies are compared. The DDC decision tree has the lowest training accuracy in average and median, but the highest testing accuracy in average and median, as well as the smallest IQR variance. This points to the stability and robustness of a DDC decision tree. Decision trees trained on numeric datasets are easily overtrained because possible splits are exhaustively searched without accounting for larger improvements by chance. This corroborates with work by Loh et al. [82], where the validity of using numerical data with the decision tree model is addressed. Therefore, this result is to be expected as the numeric dataset trains on averaged values, whereas AGNES and DDC use categorical data based on cluster results for the decision tree model.

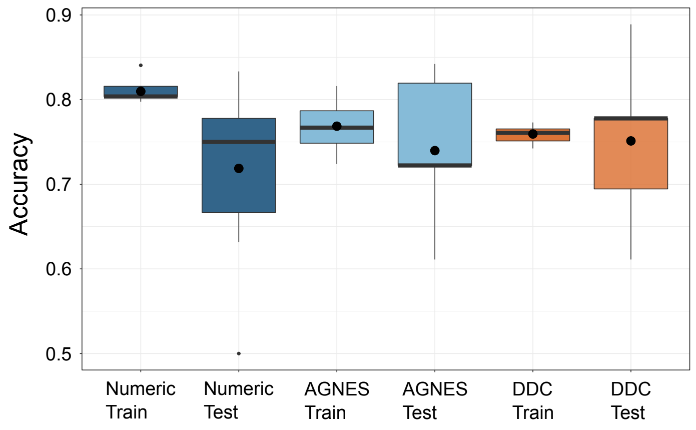


Figure 6.9: 10-fold accuracy performance comparison boxplot for discrete data, AGNES, and DDC.

Accuracy inherently depends on the data and task on hand. For a complex natural phenomenon with small datasets, the clustering and classification can be challenging. The most significant drawback in ICME classification is the lack of data; researchers are more likely to draw accurate and exhaustive conclusions from large datasets, meaning the corresponding accuracy is usually higher and the performance more stable. On top of being a small dataset, the ICME dataset of magnetic clouds are human-labeled; this means a few mislabeled samples could be affecting our results significantly. Our contribution here is the aggregation of univariate time series clustering results into a multivariate time series decision tree, which we can efficiently apply towards new data classification, while at the same time being easily interpretable.

6.4 Solar Flare Decision Tree

For solar flare decision tree experiments, we use the Prior 12 Span 06 dataset from the initial solar flare dataset. In addition to achieving high accuracy, we also generate a simple and easy to maneuver decision tree. While there are various flare classes, here we are only doing binary clustering, where an event is labeled as either a flare or a non-flare. Therefore to save training time as well as avoiding overtraining the model, we only clustered each parameter to 20 clusters. Fig. 6.10 shows the accuracy curve of the 16 parameters in the Prior 12 Span 06 dataset, each parameter is individually clustered using the DDC method. The training accuracy curve is a solid black line, and the testing accuracy curve is a dotted blue line, the five chosen parameters are highlighted with a red box. Each parameter name and the corresponding area under the accuracy curve is labeled. Each accuracy curve shows the accuracy, which is labeled on the y-axis, and the x-axis shows the corresponding cluster numbers from 2 to 20 clusters.

With testing data excluded from the training step, the consistent accuracy performance between training and testing signifies that the DDC algorithm is effective and robust. Among the top five performing parameters, parameter USFLUX, TOTUSJH, and SHRGT45 appeared in all six priors and spans, TOTUSJZ appeared in five priors and spans, MEANPOT appeared three times, and TOTPOT and MEANSHR appeared two times. Other than the MEANSHR, all the aforementioned parameters were also included in the study by Bobra et al. [70]. From the accuracy curves, we can identify knee points, where we see the most significant increase in accuracy when the cluster number is increased by 1. Since more clusters would lead to longer running time, finding the most significant accuracy increase while keeping minimal cluster number is desirable. The knee point for each corresponding parameter is the number of clusters we select to further build the decision tree.

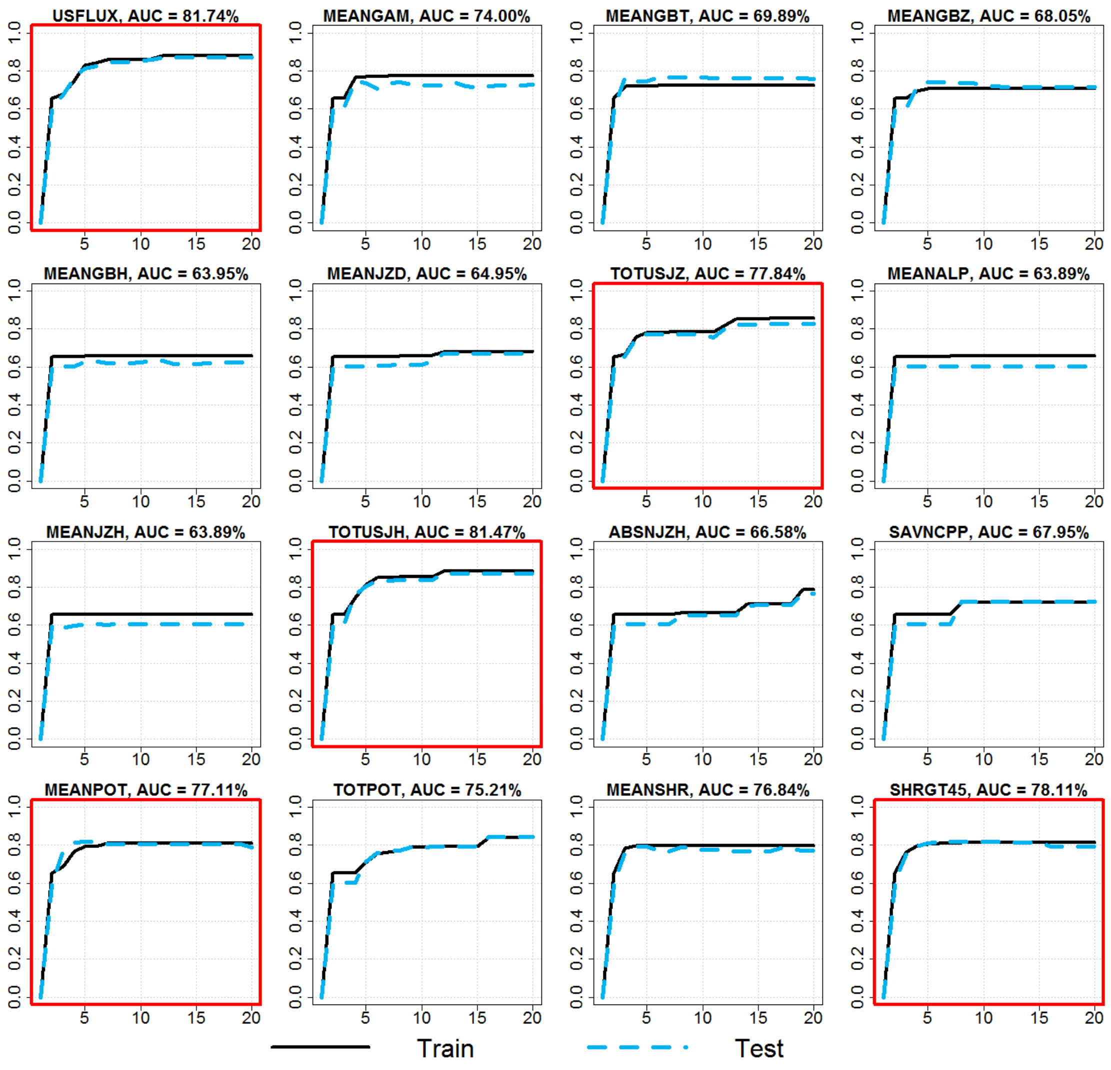


Figure 6.10: Accuracy curve for 16 parameters on both training and testing datasets of Prior 12 Span 06, with AUC labeled, and top five AUCs highlighted in red boxes.

The decision tree uses the parameters that can best separate flare and non-flare data, the nodes at the top of the tree is the parameter that can split the dataset most easily, which means that parameters on the higher levels of the tree are in a way, more informative than the parameters on the lower levels of the tree. Typically, simpler trees avoid overfitting and ensure robustness. Fig. 6.11 shows the decision tree for Prior 12 Span 06 dataset.

Each leaf node has the parameter used for a split, the number of each class event at that node. In all the decision trees as well as Fig. 6.11, the right-most and left-most leaf nodes

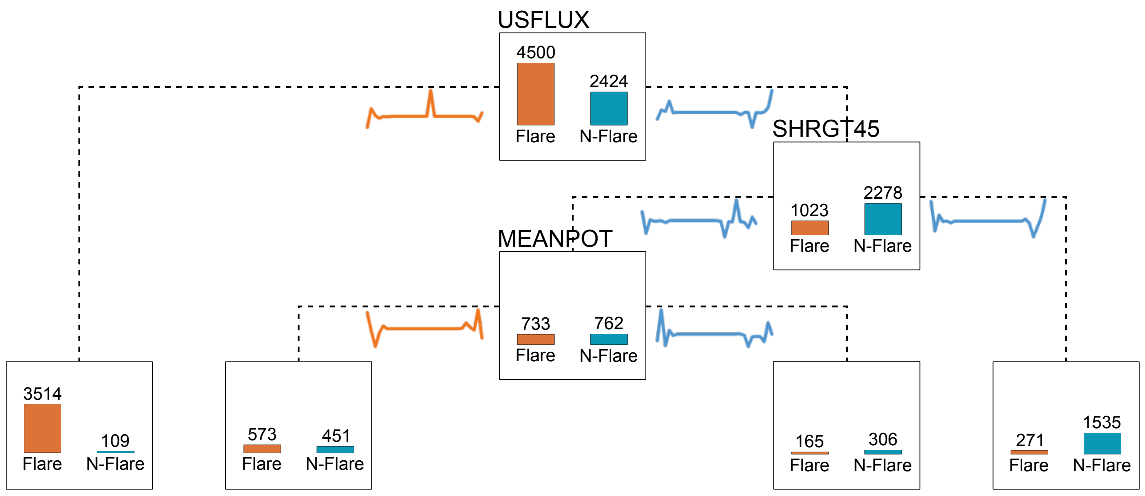


Figure 6.11: Shape-based multivariate decision tree for Prior 12 Span 06 solar flare dataset.

have the purest separation of flares and non-flares. The other leaf nodes are lower in purity, which may be a result of only performing binary clustering and not considering all the flare classes. Leaf nodes are labeled as flare if there are more flare events at that node, or labeled as non-flare if there are more non-flare events at that node. On each branch of the decision tree is the DBA of the child node cluster on the parent node parameter, providing the user with a more visual interpretation of how the shape of the time series data corresponding to the parameter at that node is utilized.

Once the decision tree model is formulated using univariate time series clustering, it is tested by trying to classify unseen flare data. During the testing process, the time series of a new event is matched to the DBA of the corresponding parameter using DTW. The purity and accuracy of each leaf node in the solar flare decision tree, as well as the placement of the testing flare data is shown in Table 6.4. Since only the parameters in the decision tree will be checked, event classification is highly efficient. This can be helpful for future flare labeling, for example, when the initial pattern matching the average time series (DBA) of the parameter within the decision tree is identified, the event would be automatically classified as flare or non-flare. With little to no human involvement, this could improve efficiency and reduce human-errors.

Table 6.4: Leaf Node Purity and Accuracy for Solar Flare Decision Tree

	Leaf 1	Leaf 2	Leaf 3	Leaf 4
Train	Events: 52% Accuracy: 97%	Events: 15% Accuracy: 56%	Events: 7% Accuracy: 65%	Events: 26% Accuracy: 85%
Test	Events: 46% Accuracy: 96%	Events: 14% Accuracy: 63%	Events: 4% Accuracy: 55%	Events: 35% Accuracy: 86%

For all priors and spans, the root nodes are either parameter USFLUX or TOTUSJZ; this corresponds to other studies on flare parameters [70]. Here either USFLUX or TOTUSJZ is able to identify around half of the entire training dataset being flare events at a 97% to 100% accuracy.

Generally, prior window 12 outperforms prior window 24; this is expected since the closer we are to a flare happening, the more likely it is to predict a flare. An accuracy of 85% for a simple structured decision tree on a large dataset is an acceptable accuracy at this stage for binary flare clustering. The reason is that flares are classified according to X-ray flux values, and therefore classes at the two ends of the spectrum, the M, X, and A, B flares are easier to classify. Whereas C class is more borderline, and depending on the specific event could be classified both ways, and would need additional effort for better placement. Based on this, we decided to look further at the shape of each flare time series through normalization.

6.5 Time Series Cluster Profile

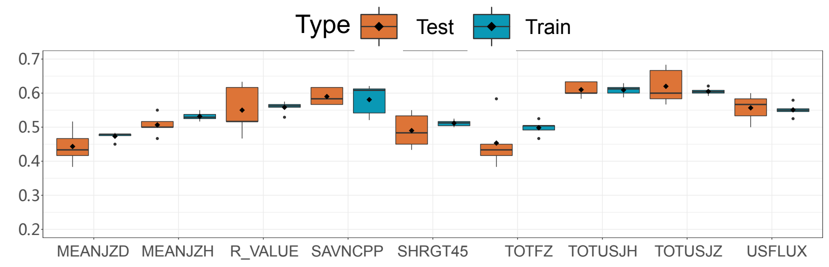
In this section, we present our findings in clustering normalized pre-flare time series data from the SWAN-SF dataset. In order to eliminate the randomness of performance, we performed a balanced 5-fold cross-validation on the curated dataset of a total of 300 C-, M-, and X-class instances. Again, for each fold, the testing data is never included in the training process to avoid bias.

Table 6.5: Nine Parameters Selected for Solar Pre-Flare Time Series Evaluation

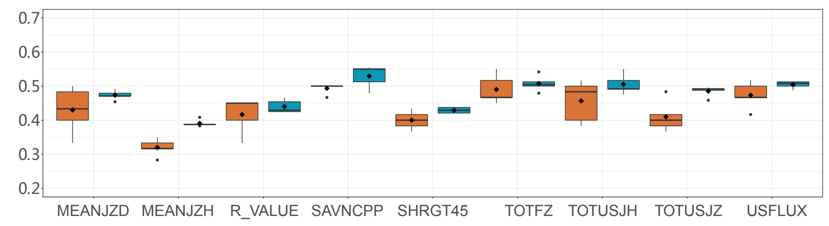
	Keyword	Description
1	MEANJZD	Mean vertical current density
2	MEANJZH	Mean current helicity
3	R_VALUE	Sum of flux near polarity inversion line
4	SAVNCPP	Sum of the modulus of the net current per polarity
5	SHRGT45	Fraction of area with shear angle $> 45°$
6	TOTFZ	Sum of z-component of Lorentz force
7	TOTUSJH	Total unsigned current helicity
8	TOTUSJZ	Total unsigned vertical current
9	USFLUX	Total unsigned flux

The first step is to obtain and compare clustering results. Distance Density Clustering is performed on the normalized time series of all the partitions on nine SHARP parameters: MEANJZD, MEANJZH, R_VALUE, SAVNCPP, SHRGT45, TOTFZ, TOTUSJH, TOTUSJZ, and USFLUX. Described briefly in Table 6.5, the nine parameters are selected by domain experts from the list of parameters discussed by Bobra et al. [70]. The different parameters are simply different measurements of solar flares, and should not hinder non-domain experts from understanding the results.

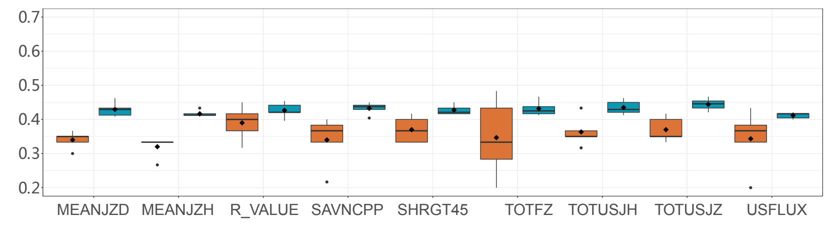
Fig. 6.12 shows the boxplot of cluster accuracy on training and testing data. In each partition for each normalization method, the data is clustered to 10 clusters. The label of a cluster is determined by the majority event labels, and multiple clusters can have the same flare class label. The x-axis shows the nine parameters, and the y-axis is the accuracy value. The accuracy at cluster 10 is computed based on the number of matches between predicted labels and actual labels for all five partitions. In other words, each boxplot contains the accuracy information from all five partitions. For each parameter, the left boxplot is the testing accuracy of each normalization method, and the right boxplot is the training accuracy of the respective normalization methods. In the unnormalized dataset, parameters R_VALUE, SAVNCPP, TOTUSJH, and TOTUSJZ have better accuracy performance. Normalization accuracy is typically lower than the original data cluster accuracy.



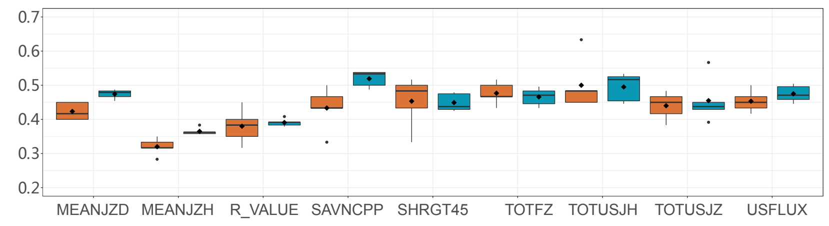
(a) Original cluster accuracy.



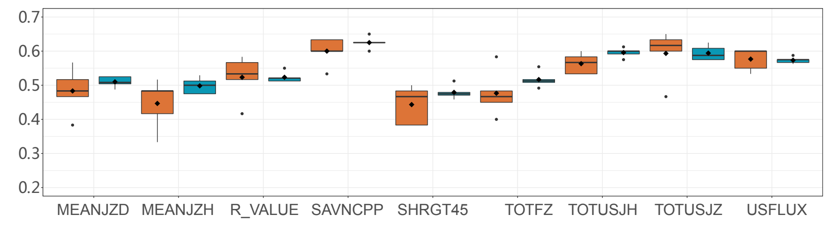
(b) Offset translation cluster accuracy boxplot.



(c) Amplitude scaling cluster accuracy boxplot.



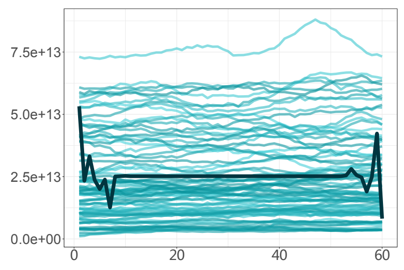
(d) Difference detrend cluster accuracy boxplot.



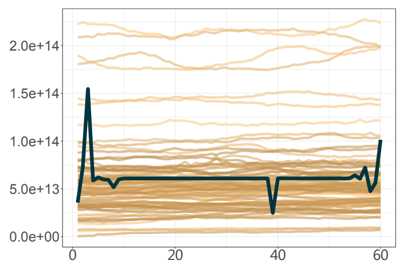
(e) Logarithmic detrend cluster accuracy boxplot.

Figure 6.12: Boxplots of cluster accuracy for (a) original data, (b) offset translation, (c) amplitude scaling, (d) detrend with difference and (e) detrend with log.

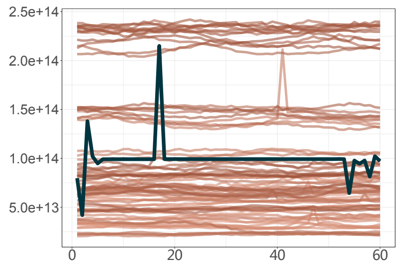
While some machine learning algorithms achieve better results from normalization, this is not the case for pre-flare time series clustering. Although we are able to obtain more shape details with normalization and generate more intuitive clusters, the resulting accuracy often declines. This is caused by the fact that when normalization is applied, only two out of three similarity elements are met, the duration and shape similarity, whereas the range value similarity is lost, and this would have a negative impact on clustering accuracy. Therefore, in our experiments, the accuracy values are only used as a reference, not a quality indicator.



(a) Pre-flare C class.



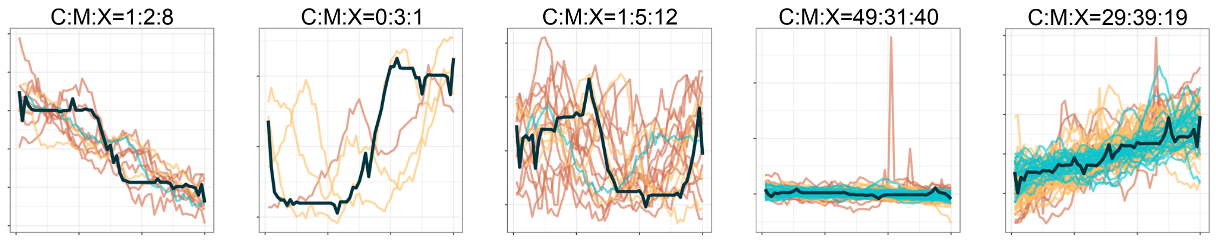
(b) Pre-flare M class.



(c) Pre-flare X class.

Figure 6.13: The original 300 pre-flare time series of (a) 100 C-class, (b) 100 M-class, and (c) 100 X-class of parameter TOTUSJZ.

Due to space limitations, we only select one parameter to demonstrate the details of further investigation. TOTUSJZ is selected as it generally has stable performance and clear cut clusters. We first look at the original, unnormalized, pre-flare time series of GOES class C, M, and X on parameter TOTUSJZ. Fig. 6.13 shows the C-, M-, and X-class pre-flare time series, along with the time series average with DBA for each class (black lines). We can observe more spikes at the beginning and end of the average time series, which is likely due to the common spikes in various time series that are being averaged. Later on, as the evens progress, there is more diversity in the time series events shapes (values), and the generally flat shape of the average time series suggests the lack of patterns to be identified. The value differences of the pre-flare time series for different classes can be easily seen. However, the shape of different pre-flare classes is not apparent; this is because the shape differences are visually suppressed by the large value differences within the same class.



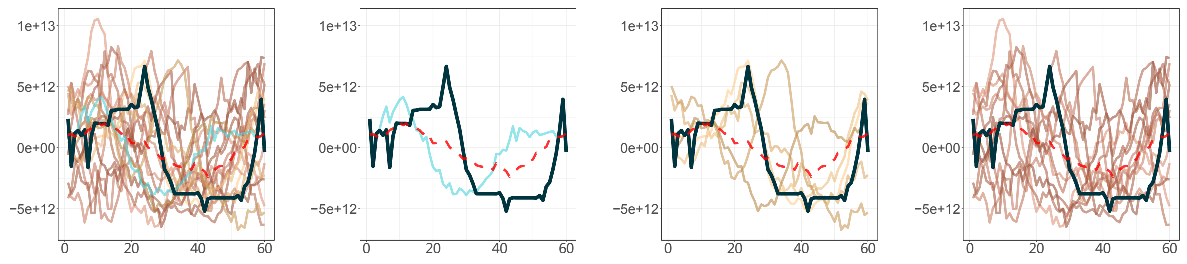
(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5

Figure 6.14: 5 clusters of parameter TOTUSJZ with offset translation.

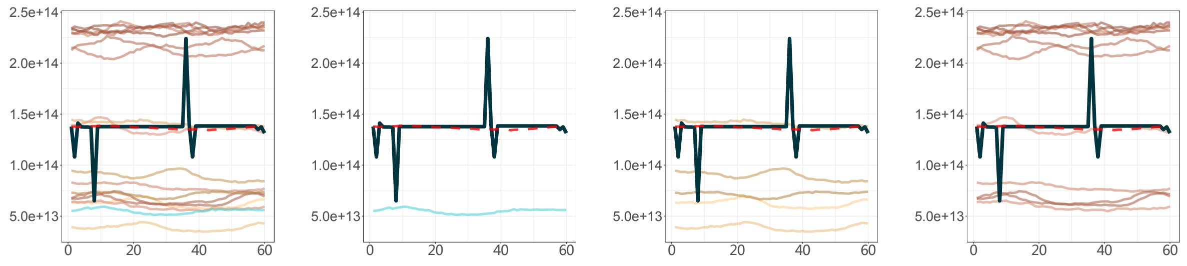
Fig. 6.14 shows the result of 5 clusters for parameter TOTUSJZ normalized with offset translation. The class ratio for each cluster is labeled above each figure, the cyan lines represent C-class, yellow lines are M-class, and orange lines are X-class. The first three clusters are smaller, and we could observe that the shape of events is highly similar within each cluster despite having different class labels. The DBA for the first three clusters is also characteristic of the cluster it is representing. For the last two clusters, the size is still quite large, and we can observe less representative DBAs.

In regards to the three aspects of similarity, since we already have duration similarity in place (i.e., all time series of SWAN-SF are of length 60), we try to emphasize shape similarity while reducing range value similarities. Fig. 6.15 shows the third cluster from Fig. 6.14. In Fig. 6.15 (a), the clustered data are all normalized with offset translation, (b),(c), and (d) shows the normalized C-, M-, and X-class pre-flare time series plotted separately. Fig. 6.15 (e) shows the original (unnormalized) time series which appear in Fig. 6.15 (a), while (f), (g), and (h) are the original time series of C-, M-, and X-class pre-flares plotted separately. Similar to Fig. 6.13, the unnormalized time series are flat, lacking shape similarities or dissimilarities for identification.

When comparing the normalized and unnormalized time series of time series from the same cluster, we can see similar shapes from different classes with a different value range. The one C-class instance in this cluster has a small value compared to other time series;



(a) Normalized cluster. (b) Normalized C. (c) Normalized M. (d) Normalized X.



(e) Unnormalized data. (f) Unnormalized C. (g) Unnormalized M. (h) Unnormalized X.

Figure 6.15: Decomposed cluster from one of five clusters from parameter TOTUSJZ. In this particular cluster, there are a total of 18 pre-flare time series, 1 C-class, 5 M-class, and 12 X-class. The normalized time series and their respective averages are in the top row, (a) the entire time series cluster, (b) C-class pre-flare, (c) M-class pre-flares, (d) X-class pre-flares. The time series in this particular cluster but in the unnormalized form is shown on the bottom row, (e) the entire cluster of unnormalized time series, (f) C-class pre-flare, (g) M-class pre-flare, (h) X-class pre-flare.

the M-class time series are all below the average line, and the X-class is demonstrating two concentrations in time series value range distribution. This suggests that different flare classes can demonstrate similar shapes, as well as the possibility of sub-classes existing within what we currently understand as one class of flares.

All the solid lines in Fig. 6.15 show the DBA time series averages, and the red dotted lines are the conventional averages. Conventional averages are simply the sum of all the instances at one time point divided by the total number of instances. When the time series are unnormalized, it is difficult for both the DBA and the conventional average to identify an intuitive representation of the original time series. In the case of normalized time series, it is still difficult for the conventional averaging technique to pick up the characteristic shape of the time series. Therefore, DBAs from normalized data is more useful in shape profile identification, and DBAs are more meaningful than conventional average generalization. It is important to note that the shape signatures we identify here are insensitive to the parameter value ranges. When we normalize, we obtain shape intensive information but miss value differences. Conversely, when we work with time series that are unnormalized, we preserve the parameter values, but overlook shape information. Therefore, the combination of both the shape information as well as the value differences would be worth investigating in the future.

7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

This thesis is a pipeline covering three main components of our current work relating to time series clustering. Our contributions include a scalable and efficient elastic measure: segDTW, which is also capable of producing intuitive mappings; a time series clustering heuristic based on the density and distance information: Distance Density Clustering, which has shown its effectiveness with datasets from various domains; and the variance evaluation of time series clusters, which avoids the early combination of inter-cluster and intra-cluster similarities. Other supporting work includes hierarchical cluster structure comparison, internal and external indices observation, and applications.

The main components can be individually applied or work collectively to analyze real-world data and provide tangible results. In application, the shape-based univariate clustering results can be aggregated with a decision tree algorithm to provide multivariate time series decisions. The shape-based decision trees are effective and robust when combined with suitable similarity measures and appropriate clustering heuristics. Another application of time series clustering is to use normalization to magnify the details of shapes and trends that may be lost due to the value disparity. By generating cluster profiles, we open the doors to the real-time classification and prediction of big data. With time series normalization, it is also possible to discover hidden patterns or sub-classes. Clustering is a broad aspect of exploratory data analysis, and therefore, much more work could be done.

7.2 Future Work

The significance of unsupervised learning, or more specifically, clustering, is its ability to analyze data with minimal given information. As such, we see clustering widely applied in various aspects, including segment analysis, outlier detection, specific and broad cluster applications. The next step for us is to advance further into time series cluster evaluation. A key component to extending our research in cluster evaluation is to perform an extensive evaluation of current evaluation methods. We are in the process of adapting more internal indices for various time series cluster evaluations. In applications, we are extending our cluster profiling work to establish a multi-level clustering heuristic that could potentially provide users with the choice of clustering precision. A clustering heuristic based on a multitude of resolutions may also provide more insight into the structure of time series clusters.

Furthermore, identifying more ways to measure time series similarity would continue to be part of our ongoing work, as any improvements, both generic or exclusive, can greatly affect both supervised and unsupervised time series data learning. With many real-world domains, the data is collected on a multi-dimensional scale, which worsens the existing issues we face with time series learning. Thus a time series specific multivariate clustering heuristic would be an invaluable addition to time series clustering. In our subsequent work, we intend to make a comprehensive evaluation of existing multivariate clustering heuristics and look into the possibility of specific and scalable multivariate time series clustering heuristics. At the root of the many issues we face with time series analysis, is the lack of understanding of time series distribution, and because time series are innately high dimensional, this is no trivial matter. Therefore recognizing the distribution of time series datasets could be the key to solving many of our problems. Although we are at the end of this thesis, the possibilities for future data exploratory endeavors are endless.

Bibliography

[1] J Todd Hoeksema, Yang Liu, Keiji Hayashi, Xudong Sun, Jesper Schou, Sebastien Couvidat, Aimee Norton, Monica Bobra, Rebecca Centeno, KD Leka, et al. The helioseismic and magnetic imager (hmi) vector magnetic field pipeline: Overview and performance. *Solar Physics*, 289(9):3483–3530, 2014.

[2] George Nagy. State of the art in pattern recognition. *Proceedings of the IEEE*, 56(5):836–863, 1968.

[3] Harold Edson Driver and Alfred Louis Kroeber. Quantitative expression of cultural relationships. *University of California Press*, 31(4), 1932.

[4] Joseph Zubin. A technique for measuring like-mindedness. *The Journal of Abnormal and Social Psychology*, 33(4):508–516, 1938.

[5] Robert Choate Tryon. Cluster analysis: correlation profile and orthometric (factor) analysis for the isolation of unities in mind and personality. edwards brother, incorporated. *Ann Arbor*, 1939.

[6] Monica Chiş, Soumya Banerjee, and Aboul Ella Hassanien. Clustering time series data: an evolutionary approach. In *Foundations of Computational, IntelligenceVolume 6*, pages 193–207. Springer, 2009.

[7] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[8] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[9] Shiming Yang, Mary Njoku, and Colin F Mackenzie. 'big data'approaches to trauma outcome prediction and autonomous resuscitation. *British Journal of Hospital Medicine*, 75(11):637–641, 2014.

[10] Daniel Mueller, VK Hughitt, M Langenberg, J Ireland, S Pagel, L Schmidt, JP Garcia Ortiz, G Dimitoglou, and B Fleck. The helioviewer project: Browsing, visualizing and accessing petabytes of solar data. In *Bulletin of the American Astronomical Society*, volume 41, page 876, 2010.

[11] Michael R Anderberg. Cluster analysis for applications. Technical report, Office of the Assistant for Study Support Kirtland AFB N MEX, 1973.

[12] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[13] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[14] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[15] Peter Ranacher and Katerina Tzavella. How to compare movement? a review of physical movement similarity measures in geographic information science and beyond. *Cartography and geographic information science*, 41(3):286–307, 2014.

[16] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

[17] Hiroaki Sakoe. Dynamic-programming approach to continuous speech recognition. In *1971 Proc. the International Congress of Acoustics, Budapest*, 1971.

[18] C Myers and L Rabiner. A level building dynamic time warping algorithm for connected word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(2):284–297, 1981.

[19] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[20] Young-Seon Jeong, Myong K Jeong, and Olufemi A Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44(9):2231–2240, 2011.

[21] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

[22] Meinard Müller. *Information retrieval for music and motion*, volume 2. Springer, 2007.

[23] Ruizhe Ma, Azim Ahmadzadeh, Soukaina Filali Boubrahimi, and Rafal A Angryk. Segmented dynamic time warping: A comparative and applicational study. In *Emerging Technologies and Applications in Data Processing and Management*, pages 1–19. IGI Global, 2019.

[24] Eamonn J Keogh and Michael J Pazzani. Derivative dynamic time warping. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–11. SIAM, 2001.

[25] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10(16), pages 359–370. Seattle, WA, 1994.

[26] Marenglen Biba and Fatos Xhafa. *Learning Structure and Schemas from Documents*, volume 375. Springer, 2011.

[27] 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.

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

[29] JB Kruskall and M Liberman. The symmetric time warping algorithm: From continuous to discrete. time warps, string edits and macromolecules, 1983.

[30] Hiroaki Sakoe and Seibi Chiba. Comparative study of dp-pattern matching techniques for speech recognition. In *1973 Tech. Group Meeting Speech, Acoust. Soc. Japan*, 1973.

[31] Lawrence R Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. PTR Prentice Hall, 1993.

[32] Byoung-Kee Yi, HV Jagadish, and Christos Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Data Engineering, 1998. Proceedings., 14th International Conference on*, pages 201–208. IEEE, 1998.

[33] Sang-Wook Kim, Sanghyun Park, and Wesley W Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 607–614. IEEE, 2001.

[34] Eamonn J Keogh and Michael J Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289. ACM, 2000.

[35] Jiaping Zhao and Laurent Itti. shapedtw: shape dynamic time warping. *arXiv preprint arXiv:1606.01601*, 2016.

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

[37] Eamonn J Keogh and Michael J Pazzani. Relevance feedback retrieval of time series data. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 183–190, 1999.

[38] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

[39] Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

[40] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[41] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[42] Leonard Kaufman and Peter J Rousseeuw. Clustering by means of medoids. statistical data analysis based on the l1 norm. *Y. Dodge, Ed*, pages 405–416, 1987.

[43] David Arthur and Sergei Vassilvitskii. K-means++: the advantages of careful seeding. In *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[44] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd*, 96(34):226–231, 1996.

[45] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. CRC press, 2014.

[46] CS Warnekar and G Krishna. A heuristic clustering algorithm using union of overlapping pattern-cells. *Pattern recognition*, 11(2):85–93, 1979.

[47] Wei Wang, Jiong Yang, Richard Muntz, et al. Sting: A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195, 1997.

[48] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of 1998 ACM SIGMOD Int. Conf. on Management of Data*, volume 27, pages 94–105. ACM, 1998.

[49] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB*, volume 98, pages 428–439, 1998.

[50] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.

[51] Vladimir Estivill-Castro. Why so many clustering algorithms: a position paper. *ACM SIGKDD explorations newsletter*, 4(1):65–75, 2002.

[52] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[53] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[54] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[55] Robert R Sokal. A statistical method for evaluating systematic relationships. *Univ. Kansas, Sci. Bull.*, 38:1409–1438, 1958.

[56] François Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping Chen, and Eamonn Keogh. Dynamic time warping averaging of time series allows faster and more accurate classification. In *2014 IEEE international conference on data mining*, pages 470–479. IEEE, 2014.

[57] Petros Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and V Vinay. Clustering large graphs via the singular value decomposition. *Machine learning*, 56(1-3):9–33, 2004.

[58] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[59] Vit Niennattrakul and Chotirat Ann Ratanamahatana. On clustering multimedia time series data using k-means and dynamic time warping. In *null*, pages 733–738. IEEE, 2007.

[60] Lalit Gupta, Dennis L Molfese, Ravi Tammana, and Panagiotis G Simos. Nonlinear alignment and averaging for estimating the evoked potential. *IEEE Transactions on Biomedical Engineering*, 43(4):348–356, 1996.

[61] Vit Niennattrakul and Chotirat Ann Ratanamahatana. Shape averaging under time warping. *2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, 2009.

[62] Waleed H Abdulla, David Chow, and Gary Sin. Cross-words reference template for dtw-based speech recognition systems. In *TENCON 2003. Conference on convergent technologies for Asia-Pacific region*, volume 4, pages 1576–1579. IEEE, 2003.

[63] Robert R Sokal and F James Rohlf. The comparison of dendrograms by objective methods. *Taxon*, 11(2):33–40, 1962.

[64] Roger Meredith, Bryan Mensi, and Marlin Gendron. Hierarchical clustering of historic sound speed profiles. In *OCEANS 2008*, pages 1–7. IEEE, 2008.

[65] Space Studies Board, National Research Council, et al. *Severe space weather events: Understanding societal and economic impacts: A workshop report*. National Academies Press, 2009.

[66] P Riley and IG Richardson. Using statistical multivariable models to understand the relationship between interplanetary coronal mass ejecta and magnetic flux ropes. *Solar Physics*, 284(1):217–233, 2013.

[67] D Du, PB Zuo, and XX Zhang. Interplanetary coronal mass ejections observed by ulysses through its three solar orbits. *Solar Physics*, 262(1):171–190, 2010.

[68] Natalia Papitashvili. Cohoweb service and data documentation, 2017. [Online; Accessed June 10, 2017].

[69] European Space Agency. Ulysses final archive user manual, 2017. [Online; Accessed June 10, 2017].

[70] Monica G Bobra and Sebastien Couvidat. Solar flare prediction using sdo/hmi vector magnetic field data with a machine-learning algorithm. *The Astrophysical Journal*, 798(2):135, 2015.

[71] Monica G Bobra, Xudong Sun, J Todd Hoeksema, M Turmon, Yang Liu, Keiji Hayashi, Graham Barnes, and KD Leka. The helioseismic and magnetic imager (hmi) vector magnetic field pipeline: Sharps–space-weather hmi active region patches. *Solar Physics*, 289(9):3549–3578, 2014.

[72] Rafal A Angryk, Petrus C Martens, Berkay Aydin, Dustin Kempton, Sushant S Mahajan, Sunitha Basodi, Azim Ahmadzadeh, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, Michael A Schuh, et al. Multivariate time series dataset for space weather data analytics. 2019.

[73] Ruizhe Ma, Azim Ahmadzadeh, Soukaina Filali Boubrahimi, and Rafal A Angryk. Segmentation of time series in improving dynamic time warping. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3756–3761. IEEE, 2018.

[74] Ruizhe Ma, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, and Rafal A Angryk. A scalable segmented dynamic time warping for time series classification. In *International Conference on Artificial Intelligence and Soft Computing (ICAISC)*, pages 397–408. IEEE, 2019.

[75] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

[76] Ruizhe Ma and Rafal Angryk. Distance and density clustering for time series data. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pages 25–32. IEEE, 2017.

[77] Ruizhe Ma, Soukaina Filali Boubrahimi, and Rafal Angryk. Time series distance density cluster with statistical preprocessing. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 371–381. Springer, 2018.

[78] Ruizhe Ma, Rafal A Angryk, Pete Riley, and Soukaina Filali Boubrahimi. Coronal mass ejection data clustering and visualization of decision trees. *The Astrophysical Journal Supplement Series*, 236(1):14, 2018.

[79] Ruizhe Ma, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, and Rafal A Angryk. Solar flare prediction using multivariate time series decision trees. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2569–2578. IEEE, 2017.

[80] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[81] Ruizhe Ma, Rafal Angryk, and Pete Riley. A data-driven analysis of interplanetary coronal mass ejecta and magnetic flux ropes. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 3177–3186. IEEE, 2016.

[82] Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica sinica*, pages 815–840, 1997.