香港浸會大學
HONG KONG BAPTIST UNIVERSITY

HKBU Scholars

DOCTORAL THESIS

Shapelet Discovery for Time Series Analysis

LI, Guozhong

Date of Award:
2021

Link to publication

General rights
Copyright and intellectual property rights for the publications made accessible in HKBU Scholars are retained by the authors and/or other copyright owners. In addition to the restrictions prescribed by the Copyright Ordinance of Hong Kong, all users and readers must also observe the following terms of use:

- Users may download and print one copy of any publication from HKBU Scholars for the purpose of private study or research
- Users cannot further distribute the material or use it for any profit-making activity or commercial gain
- To share publications in HKBU Scholars with others, users are welcome to freely distribute the permanent URL assigned to the publication

Download date: 09 Nov, 2026

HONG KONG BAPTIST UNIVERSITY

Doctor of Philosophy

THESIS ACCEPTANCE

DATE: August 23, 2021

STUDENT'S NAME: LI Guozhong

THESIS TITLE: Shapelet Discovery for Time Series Analysis

This is to certify that the above student's thesis has been examined by the following panel members and has received full approval for acceptance in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Chairman:	Prof Zhou Changsong Professor, Department of Physics, HKBU (Designated by Dean of Faculty of Science)
Internal Members:	Prof Cheung Yiu Ming Professor, Department of Computer Science, HKBU (Designated by Head of Department of Computer Science)
	Dr Huang Xin Assistant Professor, Department of Computer Science, HKBU
External Examiners:	Prof Chen Yi Henry J. Leir Chair and Professor Martin Tuchman School of Management New Jersey Institute of Technology
	Dr Lu Haiping Senior Lecturer in Machine Learning Department of Computer Science The University of Sheffield
Proxy:	Prof Ng Joseph K Y Professor, Department of Computer Science, HKBU
In-attendance:	Dr Choi Koon Kau Associate Professor, Department of Computer Science, HKBU

Issued by Graduate School, HKBU

Shapelet Discovery for Time Series Analysis

LI Guozhong

A thesis submitted in partial fulfilment of the requirements

for the degree of

Doctor of Philosophy

Principal Supervisor:

Dr. CHOI, Byron Koon Kau (Hong Kong Baptist University)

August 2021

DECLARATION

I hereby declare that this thesis represents my own work which has been done after registration for the degree of PhD at Hong Kong Baptist University, and has not been previously included in a thesis or dissertation submitted to this or any other institution for a degree, diploma or other qualications.

I have read the University's current research ethics guidelines, and accept responsibility for the conduct of the procedures in accordance with the University's Research Ethics Committee (REC). I have attempted to identify all the risks related to this research that may arise in conducting this research, obtained the relevant ethical and/or safety approval (where applicable), and acknowledged my obligations and the rights of the participants.

Signature: Li Guozhong

Date: August 2021

Abstract

Time series shapelets (or simply, shapelets) are discriminative subsequences that have been recently found effective for time series analysis, including classification and clustering. The quality of shapelets is crucial to both the accuracy and efficiency of time series analysis. However, the major of research has focused on building accurate models from some shapelet candidates rather than *discovering* high-quality shapelets. Discovery of high-quality shapelets is known to be computationally costly. Furthermore, shapelet discovery for time series analysis has a few challenges. Among the few existing work on shapelet discovery, they cannot be applied to multivariate time series classification (MTSC) since the shapelet candidates of MTSC may come from different variables of different lengths and thus cannot be directly compared. The current unsupervised shapelet discovery method cannot determine clustering quality on both univariate and multivariate time series. In this dissertation, first, we propose BSPCOVER to discover a set of high-quality shapelets for model building and then propose a matrix profile-based shapelet approach (MPS) to further improve the efficiency of shapelet discovery. Second, we propose ShapeNet for MTSC, a deep learning approach for discovering shapelets, and Autoencoder for Shapelets (AUTOSHAPE) for clustering, which takes the advantages of both shapelets and autoencoder for determining shapelets in an unsupervised manner. Experimental evaluation on well-known time series datasets (UCR and UEA archive) shows the superiority of our proposed approaches on both accuracy and efficiency. Last but not least, one of the strengths of shapelets is its interpretability. For all of our proposed

solutions, we illustrate the interpretability of shapelets with some case studies.

Keywords: Time Series Analysis, Shapelet Discovery, Classification, Clustering, Efficiency, Accuracy

Acknowledgements

This dissertation becomes a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

Foremost, I would like to express my sincere gratitude to my supervisor, Dr. CHOI Byron Koon Kau, for his continuous support of my Ph.D. study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and completing this thesis. My sincere thanks also go to Prof. XU Jianliang, Dr. Sourav S Bhowmick, Prof. Grace Lai-Hung Wong, and Dr. Kwok-Pan Chun for their encouragement, insightful comments and constructive suggestions during my Ph.D. study.

I also would like to thank my colleagues in the Database Research Group and Department of Computer Science at Hong Kong Baptist University for their direct and indirect help, for the stimulating discussions, and for all the fun we have had in the four years. In particular, I should mention Dr. ZHANG Zhiwei, Dr. HUANG Xin, Dr. LIU Qing, Dr. YI Peipei, Dr. XU Cheng, Dr. GUO Shangwei, Dr. LI Xiaoguo, Dr. ZHU Huaijie, Dr. FU Xiaoyi, Dr. JIANG Jiaxin, Mr. HUANG Jinbin, Mr. ZHANG Ce, Mr. XU Lyu, Mr. JI Yang, Mr. WANG Haixin, Mr. ZHU xuliang, Ms. SUN Longxu, Mr. SUN Zitan, Mr. LIAO Xuankun among many others.

Last but not least, I take this special occasion to thank my parents for unconditionally raising me, supporting me and providing me everything as best as they can throughout my life. Without them, I would never go so far.

Table of Contents

Declaration i

Abstract ii

Acknowledgements iv

Table of Contents v

List of Figures ix

List of Tables xiii

List of Algorithms xv

Chapter 1 Introduction 1

1.1 Background 1

1.2 Efficient Shapelet Discovery for Univariate Time Series Classification - A Heuristic Approach 3

1.3 Efficient Shapelet Discovery for Univariate Time Series Classification - A Matrix Profile Approach 7

1.4 A Shapelet-Neural Network Approach for Multivariate Time Series Classification 11

1.5 Shapelet Autoencoder for Time Series Clustering 14

1.6 Dissertation Organization 15

Chapter 2 Preliminaries and Literature Review 17

2.1 Preliminaries 17

2.2 Univariate Time Series Classification 19

2.2.1 Symbolic Representation-based Methods 20
2.2.2 Matrix Profile-based methods 20
2.2.3 Shapelet-based methods 21
2.3 Multivariate Time Series Classification 21
2.3.1 Model-based Methods 21
2.3.2 Neural Network-based Methods 22
2.4 Time Series Clustering 22
2.4.1 Autoencoder-based Methods 23
2.4.2 Shapelet-based Methods 23
Chapter 3 Efficient Shapelet Discovery for Univariate Time Series Classification - A Heuristic Approach 25
3.1 Preliminaries and Problem Statement 25
3.2 Efficient SAX Subsequence Computation 27
3.2.1 SAX Transformation 27
3.2.2 Bloom Filter and Bitmap of SAX Words 30
3.2.3 Similar SAX Word Pruning 33
3.2.4 SAX Word Cover for Model Building 38
3.2.5 p-Cover Setting 40
3.3 Experimental Results 44
3.3.1 Environment 44
3.3.2 Datasets and Parameters 45
3.3.3 Baselines 45
3.3.4 Experiments on Efficiency 47
3.3.5 Experiments on Accuracy 51
3.3.6 Experiments on Interpretability 55
3.4 Chapter Summary 56

Chapter 4 Efficient Shapelet Disocvery Univariate Time Series Classification - A Matrix Profile Approach **59**

4.1 Preliminaries . . . 59

4.1.1 Terminologies and Notations . . . 59

4.1.2 Three Issues of an MP Baseline Method . . . 60

4.2 MPS for Shapelets . . . 63

4.2.1 Shapelet Candidates Pool Generation . . . 64

4.2.2 Top-k Shapelet Generation . . . 66

4.3 Experimental Results . . . 69

4.3.1 Experimental Settings . . . 70

4.3.2 Experiments on Efficiency . . . 70

4.3.3 Experiments on Accuracy . . . 73

4.3.4 Experiments on Interpretability . . . 76

4.4 Chapter Summary . . . 77

Chapter 5 ShapeNet: A Shapelet-Neural Network Approach for Multivariate Time Series Classification **78**

5.1 Problem Statement . . . 78

5.2 Shapelet Neural Network Approach for MTS classification . . . 80

5.2.1 Multi-length-input Dilated Causal CNN (Mdc-CNN) . . . 80

5.2.2 Unsupervised Representation Learning . . . 81

5.2.3 Differentiation of the Loss Function . . . 84

5.2.4 Data Preprocessing . . . 86

5.2.5 Multivariate Shapelet Transformation . . . 89

5.3 Experimental Results . . . 93

5.3.1 Environment . . . 93

5.3.2 Datasets and Parameters . . . 93

5.3.3 Baselines . . . 94

5.3.4 Experiments on Convergence of Mdc-CNN 95
5.3.5 Experiments on Accuracy 95
5.3.6 Experiments on Interpretability 99
5.4 Chapter Summary 101
Chapter 6 Shapelet Autoencoder for Time Series Clustering **103**
6.1 Preliminaries and Problem Statement 103
6.2 Autoencoder for Shapelets (AutoShape) 104
6.2.1 Shapelet Discovery 104
6.2.2 Shapelet Adjustment 107
6.3 Experimental Results 110
6.3.1 Experimental Settings 111
6.3.2 Baselines 111
6.3.3 Experiments on Univariate Time Series 112
6.3.4 Experiments on Multivariate Time Series 119
6.4 Chapter Summary 122
Chapter 7 Conclusions **124**
7.1 Contributions 124
7.2 Future Directions 126
Appendix A Learning the Shapelets **128**
Bibliography **131**
List of Publications **142**
Curriculum Vitae **144**

List of Figures

1.1 The overview of BSPCOVER for shapelet discovery 3

1.2 Concatenations of time series (T_A and T_B) of two classes (namely, A and B) taken from the ArrowHead dataset of UCR Archive [17] . . 8

1.3 Two shapelets of class A for ArrowHead in Figure 1.2 8

1.4 The matrix profiles of (T_A, T_B) and (T_A, T_A) of Figure 1.2 – P_{AB} and P_{AA} 8

1.5 The difference between the matrix profile P_{AB} and P_{AA} of Figure 1.4 - diff(P_{AB}, P_{AA}) 9

1.6 The overview of matrix profile for shapelets (MPS) for time series classification 10

1.7 The overview of ShapeNet for multivariate time series classification 11

1.8 The overview of AUTOSHAPE for time series clustering 14

3.1 An example of PAA with $\omega = 10$ and SAX alphabet size with $|\Sigma| = 8$ from the **Beef** dataset for SAX sequence **DEFFF**$\cdots$**BBBBB** 26

3.2 The step function π of ω 29

3.3 An example of SAX transformation 30

3.4 Pruning redundant and non-discriminative SAX words (of 3 classes) using bloom filters 32

3.5 An example of (partial) constructing bitmaps of SAX words in Class 1 (C_1) 33

3.6 An example of finding SAX words for model building (The figure is best viewed in color) . . . 40
3.7 Efficiency vs p (of p-Cover) on four UCR datasets . . . 51
3.8 Accuracy of Random vs BSPCOVER in box plots on four UCR datasets 54
3.9 Accuracy vs p (of p-Cover) on four UCR datasets . . . 55
3.10 Accuracy vs iterations of BSPCOVER and ELIS on four UCR datasets 56
3.11 **ECGFiveDays** shapelets encodes hyperacute T wave . . . 57
3.12 A shapelet of **ItalyPowerDemand** highlighting the morning heating demand difference of summer and winter months . . . 58

4.1 An example of matrix profiles and the difference to show the first issue Discords as "shapelets" . . . 61
4.2 Efficiency vs Top-k (shapelet number) of BSPCOVER, BASE, MPS methods on BeetleFly . . . 71
4.3 Efficiency vs Top-k (shapelet number) of BSPCOVER, BASE, MPS methods on TwoLeadECG . . . 73
4.4 Critical difference diagram of the pairwise statistical comparison of 11 methods on the UCR Archive . . . 74
4.5 Accuracy by varying five kinds of shapelet's numbers on four UCR datasets . . . 76
4.6 Different shapelets (green) from MPS and BSPCOVER of ItalyPowerDemand highlighting the morning heating demand difference of summer (red) and winter (blue) months . . . 77

5.1 An illustration of the best match location of a subsequence in a time series . . . 78
5.2 An illustration of a multivariate time series dataset — extracted from Basicmotions [42] . . . 79

5.3 An elaboration of the Multi-length-input dilated causal Convolutional Neural Network (Mdc-CNN) 81
5.4 An example from Basicmotions of the violation of the second requirement of word2vec: subsequences that are far away but have a small distance between them 82
5.5 A comparison between our cluster-wise triplet loss (multiple positives and multiple negatives, both with intra distances) and original triplet loss (one positive and multiple negatives without intra distance) on ArticularyWordRecognition [42] 83
5.6 An illustration of the effect of training a model using the cluster-wise triplet loss function, positives are closer to each other and the anchor, negatives are closer to each other but farther from the anchor 86
5.7 An illustration of triplet sampling, black for Anchor, red for Positive samples, blue for Negative samples [best viewed in color] 89
5.8 An illustration of transforming an MTS instance into the MST representation 91
5.9 The convergences of the learning algorithm on some multivariate time series datasets 96
5.10 Triplet sampling vs random sampling of final shapelets 98
5.11 Utility-based vs random selection of final shapelets 99
5.12 Multivariate time series classification accuracy by varying 6 shapelet numbers 100
5.13 An example of multivariate shapelet transformation on Basicmotions [42] 100
5.14 An example of multivariate shapelet transformation on Atrialfibrillation [42] 102

6.1 Critical difference diagram of the pairwise statistical comparison of 15 methods on the UCR Archive 115
6.2 NMI by varying 5 numbers of shapelet on four UCR datasets [17] . . 115
6.3 An example of shapelet transformation on ToeSegmentation1 [17] . 117
6.4 An example of shapelet transformation on ItalyPowerDemand [17] . 118
6.5 The original BirdChicken dataset [17], the red lines show the shapelet S_1 corresponding area 118
6.6 An example of shapelet transformation on BirdChicken [17] 119
6.7 Critical difference diagram of the pairwise statistical comparison of 6 methods on the UEA Archive 121
6.8 NMI by varying 5 numbers of shapelet on four UEA datasets [42] . . 121
6.9 An example of shapelet transformation on one MTS dataset, called Epilepsy [42] 122

List of Tables

1.1 Shapelet discovery and model training time complexity of some shapelet-based methods [2] 4

2.1 Summary of frequently used notations in this thesis 19

3.1 Summary of frequently used notations in BSPCOVER 27
3.2 Datasets and parameters on the tested UCR ARCHIVE 46
3.3 Efficiency of BSPCOVER and related methods on UCR ARCHIVE (units is second; d and h denotes days and hours) 49
3.4 Efficiency of BSPCOVER and ELIS on UCR ARCHIVE 50
3.5 Accuracy of BSPCOVER and 10 related methods on UCR ARCHIVE . . 53

4.1 Summary of frequently used notations in MPS 60
4.2 The accuracy of the different top-k Shapelets on MP baseline, 1NN-ED, and 1NN-DTW on four datasets from UCR Archive 63
4.3 Efficiency of MPS and related methods on UCR Archive (second as unit) and the speedup 72
4.4 Accuracy of MPS and related methods on UCR Archive 75

5.1 Multivariate time series datasets information from UEA ARCHIVE . . 94
5.2 Accuracy of ShapeNet and the related methods on UEA ARCHIVE . . 97

6.1 Normalized mutual information (NMI) comparison on the UCR ARCHIVE 114

6.2 Normalized mutual information (NMI) ablation comparisons on UCR Archive 116
6.3 Normalized mutual information (NMI) comparison on the UEA Archive 120

List of Algorithms

3.1 SAX transformation for class C . . . 31

3.2 Construction of bloom filters and the SAX words' bitmaps of class C 34

3.3 Computing the weights for non-similar SAX words . . . 37

3.4 OneSaxCover: Heuristic algorithm for the weighted bitmap cover problem . . . 41

3.5 PSaxCover: Determining p-Cover SAX words . . . 42

3.6 Determining p for bspcover . . . 43

4.1 Generation of the pool for shapelet candidates . . . 66

4.2 Top-k shapelet generation . . . 69

5.1 Shapelet candidate generation . . . 87

5.2 Selection of triplet (APN) . . . 88

5.3 Multivariate Shapelet Transformation . . . 92

6.1 Shapelet candidate generation . . . 105

6.2 Determining final shapelets . . . 110

A.1 Shapelet learning [1] . . . 130

Chapter 1

Introduction

1.1 Background

Time series shapelets (or simply, shapelets) are discriminative time series subsequences that have been recently found effective for time series analysis, including time series classification and clustering. The quality of shapelets is crucial to both the accuracy and efficiency of time series analysis. We next introduce some background about the time series classification, multivariate time series classification, and time series clustering.

Time series classification has attracted considerable attention from both academia and industry. The classical approach to solving the time series classification problem is the whole series-based approach [2]. This method combines classifiers, such as 1-Nearest Neighbor (1NN), and similarity metrics, such as Euclidean Distance or Dynamic Time Warping distance.

Multivariate time series classification has been one of the most fundamental tasks of multivariate time series. However, multivariate time series classification has received much less research attention than the specific case of univariate time series classification. Various methods [2] have been proposed for univariate time series classification, and its accuracy has increased significantly when compared to some benchmark methods, such as 1 Nearest Neighbor (1-NN) with Euclidean distance

(ED) or Dynamic Time Warping (DTW) [3].

Time series clustering also has numerous applications in both academia and industry, and many research approaches for solving time series clustering have been proposed. The classical approaches to solving the time series clustering problem can be categorized as whole series-based, feature-based, and model-based [4]–[8]. These approaches use the raw time series themselves, feature extraction, and model parameter transformation and then apply K-means, DBSCAN, or other clustering algorithms.

A recent trend is to find small patterns that represent classes of time series. Among these patterns, shapelet-based methods (*e.g.*, [1], [9]–[13]) have repeatedly demonstrated superior accuracy. Intuitively, shapelets can be understood as discriminative subsequences that maximally represent classes of time series. Recent research has combined shapelets with learning approaches (*e.g.*, [1], [9]) to learn a few shapelets that can distinguish time series of different classes.

The two key steps of shapelet-based methods are *shapelet discovery* and *model building*. Recent research on shapelets has given much attention on accurate model building. In contrast, this thesis shows that *shapelet discovery not only significantly improves the efficiency of model building but also often improves the accuracy of the model built* on time series classification, multivariate time series classification, and time series clsutering.

In this dissertation, we study four problems, namely (i) efficient shapelet discovery for time series classification, (ii) a shapelet-neural network approach for multivariate time series classification, and (iii) shapelet autoencoder for time series clustering, and (iv) a matrix profile approach for discovering shapelets on time series classification.

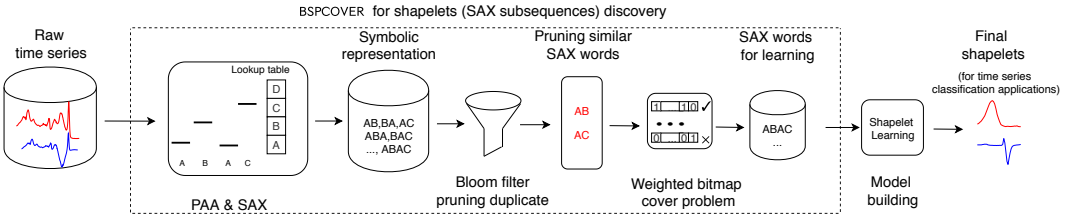


Figure 1.1: The overview of BSPCOVER for shapelet discovery

1.2 Efficient Shapelet Discovery for Univariate Time Series Classification - A Heuristic Approach

The fundamental challenge of shapelet discovery is to efficiently discover high-quality shapelets for model building. First, since a shapelet can be any subsequence, whose length is smaller than or equal to the length of raw time series, the number of candidates is large. Given an instance of a time series, the number of shapelet candidates is, however, quadratic to its length. Second, the shapelets should be discriminative enough to classify time series [14]. Shapelet discovery is worthwhile because non-discriminative subsequences harm both the efficiency and accuracy of model building. Further, subsequences that discriminate similar time series instances of one class into different classes do not improve the accuracy but also adversely affect the efficiency of model building. Third, due to efficiency issues in model building for some large datasets, only a relatively small set of candidates are selected. For example, only shapelets of three different lengths are used in [9]. It is technically intriguing to systematically select candidates for efficient model building.

Previous shapelet-based methods can be divided into three categories. **I.** Brute force algorithms (*e.g.*, [15]) exhaustively search shapelets from raw time series sequences. Information gain is adopted to measure the quality of shapelet candidates. The time complexity is $O(M^2N^4)$, where M is the number of time series instances and N is the length of time series. **II.** Some previous studies (*e.g.*, learning time-series shapelets (LTS) [9] and shapelet transformation (ST) [16]) simply set a few

Table 1.1: Shapelet discovery and model training time complexity of some shapelet-based methods [2]

Method	FS	ST & COTE	LTS	ELIS	BSPCOVER
shapelet discovery	$O(MN^2)$	negligible	negligible	$O(MN^2 + MN\log(MN))$	$O(MN^2)$
Model building	-	$O(M^2N^4)$	$O(IMN^2)$	$O(IMN^2)$	$O(IMN^2)$

fixed ratios of the raw time series lengths as the final shapelet lengths and hence the number of shapelets for model building. These ratios are sensitive to both datasets and algorithms. For example, the number of shapelets for the dataset **Beef** [17] with a length of 470 is 71 $(0.15 \times N)$ for LTS and 235 $(0.5 \times N)$ for ST. The complexity is $O(IMN^2)$, where the learning approach LTS [9] is adopted and I is the iteration number. From the experiments on [1] on **Beef**, LTS takes *more than one week* to learn a model on training data and do the classification on test data [1]. **III.** The state-of-the-art approaches SAX-VSEQL, SAX-VFSEQL [18], and ELIS [1] exploit symbolic representations of time series. The symbolic representation methods, such as Piecewise Aggregate Approximation (PAA) [19], and Symbolic Aggregate approXimation (SAX) [20] can reduce the dimensionality of the original time series. A technique is derived from TF-IDF to measure the quality of the candidates for ELIS, which can still be costly. ELIS takes *over six hours* to learn a model and do classification for **Beef**.

In Chapter 3 of this thesis, we propose *an efficient shapelet discovery method, called* BSPCOVER*, to discover high-quality shapelets* for model building. An overview of BSPCOVER is presented in Figure 1.1.

The first step of BSPCOVER follows existing work on shapelets that simply applies symbolic representation methods, specifically PAA [19] and SAX [20], to reduce the dimensionality of raw time series. In particular, they transform raw data into discrete representations (SAX subsequences, a.k.a SAX words). A sliding window method is adopted to generate numerous SAX words. The core of BSPCOVER then

[1]The experiment conducted on a Ubuntu PC with Intel(R) Xeon(R) CPU E5-2403 v2 Processor XS(8-core 1.80GHz CPU), 64GB memory and 5TB disk space.

consists of three main steps. **I.** BSPCOVER derives a bloom filter for each class of time series to efficiently prune the same SAX words that also exist in other classes. Such SAX words have less discriminative power and only deteriorate overall efficiency and accuracy. Since bloom filters do not produce false negatives, the SAX words of a class that do not exist in the bloom filters of other classes are definitely not in those other classes. Such SAX words can be candidates for model building. **II.** A non-metric distance-based pruning is proposed for the similar SAX words, which eliminates the effect of similar SAX words that exist in all classes. Because the distance between the raw time series violates triangle inequality, an intrinsic property of time series, the discrete representation of time series inherits this property. There are two situations where the similar SAX words are pruned. One is the similar SAX words exist in different classes, they are all pruned. The other is the similar SAX words in the same class, only one representative word may be kept for further processing. **III.** For each set of similar SAX words of a class, we count their term frequency, as weight. Then, we design weighted bitmap structures of SAX words to quantify the quality of shapelet candidates, which is utilized to discover a set of SAX words that have the maximal weight and represent all the instances in each class. The shapelet selection problem is then formalized as a weighted bitmap cover problem, which can be reduced from the classical weighted set cover problem. We propose a heuristic algorithm to solve this problem. The complexity of the algorithm is $O(mn^2)$ only.

Finally, to complete a TSC solution, we compute the discriminative shapelets from the candidates as follows. We transform the SAX words back to the raw representation of time series. Existing learning methods can be applied for modifying the selected shapelets to improve the accuracy further. In this thesis, we utilize LTS [9] to build one-vs-rest classifiers.

We have conducted extensive experiments by running current state-of-the-art methods and our implementation of BSPCOVER on well-known time-series datasets.

We have obtained consistent results as they are reported from the paper, unless stated otherwise. We note that BSPCOVER is faster than most shapelet-based methods, and its accuracy remains competitive with collective of transformation-based ensembles (COTE) [21], which is by far the best classifier today, according to a recent comprehensive survey of TSC [2].

We summarize the shapelet discovery and model building time complexity of some shapelet-based methods in Table 1.1. (While the complexities of FS are low, its accuracy is lower than other methods.) Model building of other methods is more costly than shapelet discovery. Due to shapelet discovery, I of BSPCOVER can often be 10 times smaller that of LTS and ELIS. Shapelet discovery makes model building converge much faster in practice.

The contributions of BSPCOVER are summarized as follows:

- We propose an approach, BSPCOVER, to discover discriminative shapelets efficiently for classifier model building for TSC. Bloom filters are adopted and a non-metric distance measure is proposed to prune non-discriminative shapelets.
- We design a data structure of SAX words, namely weighted bitmap, to measure the quality of SAX words. Then, we formalize a weighted bit cover problem based on this structure.
- The SAX words selection problem is reduced to the classical weighted set cover problem by the weighted bitmap structure, and a heuristic algorithm is introduced for discovering a set of SAX words to build the classifier model.
- Extensive experiments on UCR datasets (UCRARCHIVE) for TSC studies [17] verify that our proposed approach is significantly more efficient and yet competitive in terms of accuracy when compared to existing shapelet-based methods.

1.3 Efficient Shapelet Discovery for Univariate Time Series Classification - A Matrix Profile Approach

In this section, we introduce shapelets with matrix profile for time series classification. Univariate time series classification is one of the most fundamental analysis of time series data. UTS classification has attracted substantial research attentions, such as [2], [22]–[26]. The traditional approaches to solving the UTS classification problem can be classified into four categories, namely, whole series-based, intervals-based, dictionary-based, and model-based [2]. Raw data analysis, feature extraction, frequency of repetition of subsequences, and generative model, among other detailed methods, have been employed in these approaches. Recently, Ye et al. [14] have proposed shapelets, which can be intuitively considered as discriminative subsequences that maximally represent classes of the time series. Shapelet-based methods (*e.g.*, [1], [9], [14], [16], [27]) have repeatedly found superior in solving UTS classification. Interested readers may refer to an excellent survey [2].

Shapelets are discriminative time series subsequences originally proposed for UTS classification [14]. Shapelets can be readily interpreted by human as they are subsequences themselves. We illustrate an example of raw time series dataset, called ArrowHead from left part of Figure 1.2, and its shapelets in Figure 1.3. Lines et al. propose shapelet transformation [16] for discovering shapelets to represent the original time series into a new space. Some classic classification methods, such as Nearest Neighbor, Naive Bayes, and SVM, can then be applied to the transformed data to do the classification. Grabocka et al. [9] introduce a logistic regression function to learn near-to-optimal time series shapelets for UTS classification, called LTS. The classification accuracy of the UCR Archive [17], a well-known benchmark of time series datasets, is thus increased significantly. However, there are at least two reasons for LTS's inefficiency. The number of shapelet candidates is large, and a few

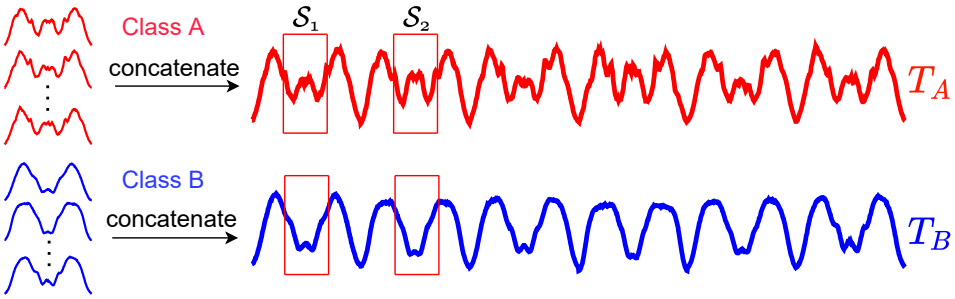


Figure 1.2: Concatenations of time series (T_A and T_B) of two classes (namely, A and B) taken from the ArrowHead dataset of UCR Archive [17]

Figure 1.3: Two shapelets of class A for ArrowHead in Figure 1.2

parameters, such as shapelet number and length, are explored manually for building a model. Currently, [27], [28] reported that the efficiency of computing shapelets can be significantly improved.

Recently, matrix profile, a data structure to annotate time series, has been proposed for computing the time series motifs and discords. One of the strengths of matrix profile methods is that it can discover the time series motifs and discords both *efficiently* and *accurately* [29]–[32]. However, few matrix profile-based shapelet methods for UTS classification have been proposed. Yeh et al. [29] sketched a baseline method, concatenating all the time series instances of each class (*e.g.*, class A and B in Figure 1.2), and then computing the corresponding matrix profile, as shown in

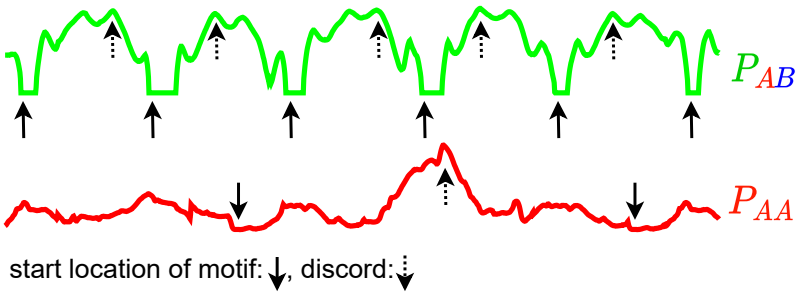


Figure 1.4: The matrix profiles of (T_A, T_B) and (T_A, T_A) of Figure 1.2 – P_{AB} and P_{AA}

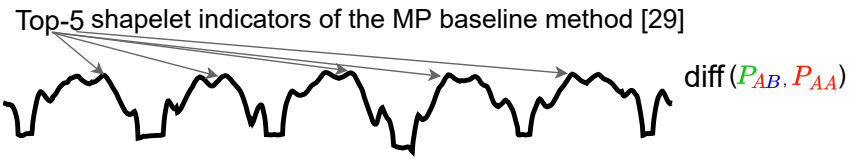


Figure 1.5: The difference between the matrix profile P_{AB} and P_{AA} of Figure 1.4 - diff(P_{AB}, P_{AA})

Figure 1.4. It has been argued [29] that the large differences (in Figure 1.5) of the matrix profiles are indicators of high-quality shapelet candidates. The intuition is that a discriminative pattern that is present in, say, class A, but not class B, results in a big gap in the values in the matrix profiles.

There are however three main issues of such a method for discovering shapelets, which have detrimental effects on the quality of the shapelets (to be detailed in Section 4.1.2). In a nutshell, the indicator may denote a situation where the shapelet candidate may be not only a discord in class B but also a discord in class A and has a large difference in the matrix profiles. This situation contradicts the definition of shapelets [14], [27] (issue 1) and can subsequently confuse shapelets determination and the classifier built from these shapelets (issue 2). To somewhat address this issue, one may pick top-k shapelets. If there are sufficiently many shapelets, the classifiers might be effective. However, it is known that the top-k shapelets can be similar (issue 3) to each other and not the most effective for solving UTS classification.

Contributions. This is *the first work to take the advantages of both shapelet-based methods and matrix profile* for UTS classification, called *multiple matrix profile*, namely MPS, which addresses the above-mentioned issues. An overview of MPS is presented in Figure 1.6.

First, different from the previous work [29] (MP baseline) to concatenate all the time series instances into one, we exploit the matrix profile on the time series instance in a bagging way [33]. Some time series instances are randomly selected from the training set to concatenate into one long time series for the shapelet candidates, which is more robust than MP baseline. In each sample, we calculate the motifs and

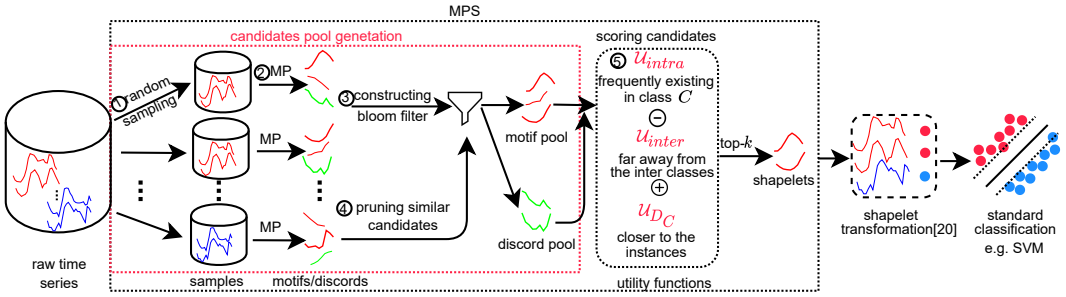


Figure 1.6: The overview of matrix profile for shapelets (MPS) for time series classification

discords for further selecting the top-k shapelets. SAX technique [34] and bloom filters are also employed for the initial candidate pruning. It is worth mentioning that the concatenating of MP baseline method [29] can be viewed as an extreme case of MPS, where all the time series instances from one class are placed into one sample.

Second, three utility functions are proposed to score the shapelet candidates from three different perspectives, namely the intra class, inter classes, and time series instances from the same class. As the name suggests, the intra class function scores the motif candidates from the same class. The motifs and discords from the inter classes are utilized to eliminate the shapelet candidates of low quality. For the shapelet determination confusion, the time series instances from the same class can further contribute to the final shapelets.

Finally, the final shapelets are employed for shapelet transformation rather than directly using shapelets to do the classification. The original time series are transformed by the final shapelets and the transformed representations are passed to construct a classification model, such as SVM, to perform the classification task.

We conduct extensive experiments on the UCR time series archive. The comparison with MP baseline and BSPCOVER proves the superiority of MPS in terms of efficiency. The accuracy shows that MPS is ranked $2nd$ among all the 11 compared to the state-of-the-art methods. We note that MPS performs the best in 10 datasets out of 46 datasets. We illustrate one case of ItalyPowerDemand to present the interpretability of discovered shapelets.

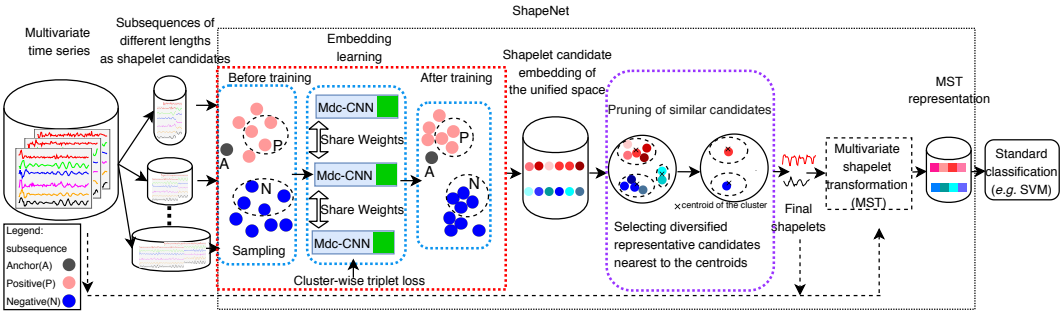


Figure 1.7: The overview of ShapeNet for multivariate time series classification

1.4 A Shapelet-Neural Network Approach for Multivariate Time Series Classification

In this section, we introduce shapelets for multivariate time series. Multivariate time series (MTS), containing multiple observations at each timestamp, are ubiquitous in many applications, ranging from astronomy, biology, geoscience, and smart cities, to health care, human action recognition, marketing, and other scientific and social domains. For example, data from electroencephalography (EEG) and magnetoencephalography (MEG) are standard multivariate data that have a wide range of applications in medicine, neurology, and psychology.

Some related studies on improving MTS classification accuracy are presented in Chapter 2. In particular, *shapelets* [14] are short discriminative time series subsequences. The effectiveness of shapelet-based classifiers of UTS classification has been proven by many related studies in the last decade, *e.g.*, logical shapelets [35], fast shapelets [10], learning shapelets [9] and dynamic shapelets [36]. Their efficiency has improved significantly recently [27], [28]. Importantly, shapelets themselves are intuitive, and the distances between shapelets and time series from different classes indicate significant differences in the classes. To integrate shapelets with standard classifiers, such as SVM and the Naive Bayes classifier, shapelet transformation [16] has been proposed.

Challenges. A shapelet-based approach for MTS classification is in its infancy, however. Few shapelet-based methods for MTS classification have been introduced [37][38]. The challenges of a shapelet approach for MTS classification can be listed as follows.

- First, multivariate time series, of course, have multiple variables. Shapelet candidates can be *voluminous* and *heterogeneous*. Exhaustive searches of shapelets [37][38] can be inaccurate.
- Second, shapelet candidates of different variables can be of different lengths, and such shapelets are *hard to compare*. With excessive candidates, it is not clear how to select the discriminative ones for classification.
- Third, most existing studies take a black-box approach. Few methods provide interpretable results for understanding and explaining the classification. It is crucial that the MTS classification approach maintains the interpretability of shapelets.

Contributions. In this thesis, we propose a new shapelet-neural network approach for the MTS classification problem, called ShapeNet, to address the challenges mentioned above. An overview of ShapeNet is presented in Figure 1.7. The benefits of ShapeNet are twofold, namely, *accuracy improvement* and *interpretable classification results*.

First, we propose the *Multi-length-input dilated causal Convolutional Neural Network (Mdc-CNN)*, which enhances Dc-CNN [39], to embed shapelet candidates of different lengths and different variables into a unified space (shapelet embedding). We adopt *dilated convolution* [40], which enables an exponentially large receptive field of the sequence for handling long-term dependencies without an explosion of model complexity. *Causal convolution* [40] is adopted for convolving only the time before the current time, which ensures that no future value impacts the current value. In addition, we propose a *cluster-wise triplet loss function* for training Mdc-CNN that considers intra/inter cluster metric learning for accelerating convergence and

improving stability. Our cluster-wise triplet loss not only takes multiple positive samples and multiple negative samples as input, but also calculates the distance between them. In comparison, the previous triplet loss [41] only involves one positive sample and one negative sample. Our loss function is more robust for faster determination of shapelet embedding and convergence (see Figure 5.5). To the best of our knowledge, this work is the first to use a neural network to discover shapelets in MTS.

Second, we avoid directly feeding numerous shapelet candidates (encoded by the embedding learning using Mdc-CNN) to build a classifier. We first cluster the shapelet candidate embeddings. We then propose a *utility function* to select top-k candidates which are close to the centroid of a large cluster and far away from other cluster centroids, which gives us *representative and diversified* final shapelets.

We then adopt *multivariate shapelet transformation* (MST), which is first formally defined. Specifically, given a multivariate time series, we compute its distance(s) to the selected shapelet(s) of the same variable to obtain a MST representation.

In all, ShapeNet learns the variable-length time series subsequences of different variables into the unified embeddings, where ShapeNet captures the interactions among different variables in MTS.

Finally, because of the MST representation, we can readily learn a classification model. In this thesis, we adopt linear SVM, which allows us to visualize how the shapelets of different variables separate the time series of different classes in the case studies.

We conduct experiments on UEA MTS Archive [42]. The results show that ShapeNet is the best of the baselines and the state-of-the-art methods in terms of accuracy. We note that ShapeNet gives the best performance in 14 datasets out of 30 datasets. We present two cases of human action recognition and ECG data, to illustrate how do the shapelets give insights into classification.

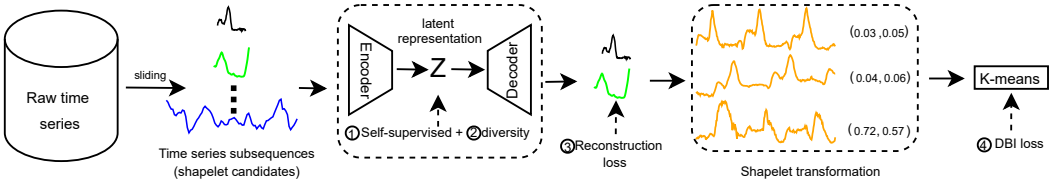


Figure 1.8: The overview of AUTOSHAPE for time series clustering

1.5 Shapelet Autoencoder for Time Series Clustering

In this section, we introduce shapelets for time series clustering. In the seminal work on shapelets [14], shapelets are discriminative time-series subsequences, providing interpretable results, proposed for time series classification. The unsupervised shapelet method (a.k.a. U-shapelets) [11] applies shapelets for time series clustering. Scalable U-shapelets [12] have been proposed for the efficient discovery of shapelets. To further improve cluster quality, Zhang et al. introduced an unsupervised shapelet learning model [13] whose method is similar to the learning shapelets [9] on time series classification. We proposed the ShapeNet framework to discover shapelets for multivariate time series classification [43].

Recently, autoencoder-based methods (*e.g.*, DEC [44], IDEC [45]) have been applied to the clustering problem and produce effective results. Most of these methods utilize an autoencoder network to learn general representations of whole time series. Different from these previous autoencoder-based work, we learn a unified representation for variable-length time series subsequences (shapelet candidates) of different variables. When all the shapelet candidates are embedded into the same latent space, it becomes simple to measure the similarity among the candidates and to further determine shapelets for clustering. In addition, because of the autoencoder approach, shapelets are no longer restricted to real-time series subsequences.

In this thesis, we propose a novel autoencoder-based shapelet approach for the time series clustering problem, called AUTOSHAPE. An overview of AUTOSHAPE is presented in Figure 1.8. To the best of our knowledge, this is the first study to take

the advantages of both shapelet-based methods and autoencoder-based methods for time series clustering.

Four objectives are specially designed for learning the final shapelets. First, self-supervised loss learns general unified embeddings of shapelet candidates. Specifically, we employ a *cluster-wise triplet loss* [43], which has been shown effective for representing time series. Second, we introduce a novel *diversity loss* to learn top-k candidates after clustering all the embeddings. The learned candidates, closest to the centroid of a cluster, are of high quality that have two characteristics: they yield large clusters, and the clusters are distant from each other. Third, we decode the selected embeddings through the *reconstruction loss* to obtain decoded shapelets. Such shapelets maintain the shapes of the original time series for human interpretation. Then, the original time series are transformed with respect to the decoded shapelets. The new transformed representations are passed to construct a clustering model (*e.g.*, K-means). Fourth, after achieving the clustering result, the *Davies–Bouldin index* (DBI) [46] is calculated to adjust the shapelets.

We conduct comprehensive experiments both on the univariate time series (UCR Archive [17]) and the multivariate time series (UEA Archive [42]). The results show that in terms of normalized mutual information (NMI), AutoShape is the best of all 14 and 5 state-of-the-art methods compared on univariate and multivariate time series (uts and mts), respectively. We note that AutoShape performs the best in 15 out of 36 uts datasets and 24 out of 30 mts datasets. Furthermore, an ablation study shows its effectiveness on self-supervised loss, diversity loss and DBI objective. We present three cases from UCR archive (human motion recognition, image, and sensor data) and one case from UEA archive (EEG of human activity recognition), to illustrate the interpretability of the learned shapelets for clustering.

1.6 Dissertation Organization

With respect the works discussed earlier, the rest of the dissertation is organized as

follows. Chapter 2 reviews existing studies. Chapter 3 presents the scalable and efficient time series classification. Chapter 5 studies a shapelet-neural network approach for multivariate time series classification. An autoencoder network, AutoShape, to learn the representations of time series subsequences (namely, shapelet candidates) is investigated in Chapter 6. A matrix profile approach for efficient shapelet discovery on time series classification is introduced in Chapter 4. Finally, the dissertation is concluded in Chapter 7.

Chapter 2

Preliminaries and Literature Review

In this chapter, we first present some preliminaries and summarize the notations and their meanings Section 2.1. We then survey related work on univariate time series classification Section 2.2, multivariate time series classification Section 2.3, and time series clustering Section 2.4.

2.1 Preliminaries

In this section, we present some preliminaries and summarize the notations and their meanings in Table 2.1.

Definition 2.1. Time series T. A time series T is an ordered-value sequence $T = (t_1, t_2, \cdots, t_n, \cdots, t_N)$, where N is the length of T, t_n is the value observed at timestamp n. □

Definition 2.2. Time series dataset D. A time series dataset D is a set of time series T_m with $C_m = \mathsf{label}(T_m), m \in [1, M]$, where $C_m \in \mathcal{C}$ is the class label of the dataset, $\mathcal{C} = \{0, 1, 2, \cdots, |\mathcal{C}| - 1\}$, and $|\mathcal{C}|$ denotes the number of classes. □

Definition 2.3. Multivariate time series $\mathbb{T}$. A multivariate time series $\mathbb{T}$ is $(T^1, T^2, \cdots, T^V)$, where V is the number of variables (a.k.a observations) in $\mathbb{T}$. T^v is an ordered-value sequence of length N, $T^v = (t_1^v, t_2^v, \cdots, t_N^v)$. For each specific timestamp $n \in [1, N]$, $t_n \in \mathbb{R}^V$ consists of V variables $\{t_n^1, t_n^2, \cdots, t_n^V\}$. □

Definition 2.4. Multivariate time series dataset $\mathbb{D}$. A multivariate time series dataset $\mathbb{D}$ contains M multivariate time series. Denote the number of time series in $\mathbb{D}$ as M. $\mathbb{T}_m$ is a multivariate time series with $C_m, m \in [1, M]$, where $C_m \in \mathcal{C}$ is the class label of the dataset, $\mathcal{C} = \{0, 1, 2, \cdots, |\mathcal{C}| - 1\}$, and $|\mathcal{C}|$ denotes the number of classes. □

Definition 2.5. Subsequence. Given a time series T, a subsequence of T is, $T(a, b) = (t_a \cdots, t_b)$, where $1 \leq a \leq b \leq N$, a and b are the beginning and ending positions to the subsequence, respectively. □

Definition 2.6. Distance between two sequences [1]. The distance of sequence T_p of the length N_p and T_q of the length N_q is denoted as (w.l.o.g. assuming $N_q \geq N_p$),

$$\mathsf{dist}(T_p, T_q) = \min_{j=1,\cdots,N_q-N_p+1} \frac{1}{N_p} \sum_{l=1}^{N_p} (t_{q,j+l-1} - t_{p,l})^2. \tag{2.1}$$

□

Definition 2.7. Shapelet S_i [14]. A shapelet S_i of the length L in class C_j, where $C_j \in \mathcal{C}$, is a time series subsequence, which can represent class C_j and discriminate C_j from other classes, *i.e.*, $\mathcal{C} \setminus \{C_j\}$. That is, $\mathsf{dist}(T_j, S_i)$ is smaller for all $\mathsf{label}(T_j) = C_j$, whereas $\mathsf{dist}(T_{j'}, S_i)$ is larger for all $\mathsf{label}(T_{j'}) = \mathcal{C} \setminus \{C_j\}$. □

According to Eqn. 2.1, the distance between the j-th time series T_j and a shapelet candidate S_i of length L_{S_i} is defined as follows:

$$d_i^j = \mathsf{dist}(T_j, S_i) \tag{2.2}$$

□

Definition 2.8. Shapelet transformation [16]. Shapelet transformation is a method to transform a time series T_j, w.r.t. the shapelets $\mathcal{S} : \{S_1, \cdots, S_k\}$, into a new data space (d_1^j,$\cdots$,d_k^j), where $d_i^j = \mathsf{dist}(T_j, S_i)$, and $\mathsf{dist}(T_j, S_i)$ is the distance between T_j and a shapelet S_i in $\mathcal{S}$. □

Table 2.1: Summary of frequently used notations in this thesis

Notation	Meaning
T	a time series $(t_1, t_2, \cdots, t_i, \cdots, t_N)$, where t_i is the i-th value in T and N is the length of T
$T_{a,b}$	a subsequence $T_{a,b}$ of T, $(t_a, \cdots, t_b)$, where $1 \leq a \leq b \leq N$, a and b, the beginning and ending positions
D	a time series dataset $(T_1, T_2, \cdots, T_M)$, where M is the number of time series in D
V	the number of variables/observations
$\mathbb{T}$	a MTS $\mathbb{T} = (T^1, T^2, \cdots, T^v \cdots, T^V)$, where $T^v = (t_1^v, t_2^v, \cdots, t_i^v, \cdots, t_N^v)$
$\mathbb{D}$	a MTS dataset $(\mathbb{T}_1, \mathbb{T}_2, \cdots, \mathbb{T}_M)$, where M is the number of MTS in $\mathbb{D}$
$\mathcal{C}$	the label set, $\{0, 1, 2, \cdots, \lvert\mathcal{C}\rvert - 1\}$
D_C	C is the label in $\mathcal{C}$; and for all $T \in D_C$, the label of T is C
$\mathcal{S}$	the shapelets
UTS	univariate time series
MTS	multivariate time series

For simplicity, we also use $\mathsf{dist}(T_j, \mathcal{S})$ to denote the distance between T_j and $\mathcal{S}$, *i.e.*, $\mathsf{dist}(T_j, \mathcal{S}) = (d_1^j, \cdots, d_k^j)$. Shapelets can be applied to reduce the dataset by using the transformation in Def. 2.8. The number of data values of the dataset D is reduced from $m \times n$ to $m \times k$, where k is often much smaller than n.

2.2 Univariate Time Series Classification

There are a large number of existing methods for time series classification problem. Interested readers may refer to an excellent review paper [2]. However, the efficiency of time series classification has not been the primary concern of research. In this section, we mainly review three representative methods, namely *symbolic representation*-based methods, *matrix profiel*-based methods, and *shapelet*-based methods.

2.2.1 Symbolic Representation-based Methods

There are a few symbolic representation-based methods such as PAA [1], SAX [18], [34], Symbolic Fourier Approximation (SFA) [47], DFT and other discretization techniques. Another notable example is the dictionary-based method. These methods approximate and reduce the dimensionality of series through transforming raw data into representative words. Bag-of-Patterns (BOP) [48] builds a classifier from SAX, calculating the Euclidean distance with the BOP representation of a new instance to determine its class. Another approach, SAX-VSM [49], adopts SAX techniques, proposes a vector space model (VSM), and uses TF-IDF to rank time series patterns. Le et al. [18] combined various variable-length bag-of-symbolic-words representations and an efficient linear sequence learning approach (SAX-VSEQL [50]) for efficient time series classification.

2.2.2 Matrix Profile-based methods

Matrix profile [29] is an annotation series, where each element is the nearest neighbor distance of the subsequences (see Section 4.1 for background). Recently, matrix profile methods (*e.g.*, [29]–[32]) have been applied to the time series data mining (*e.g.*, motif discovery and anomaly detection) and produce both efficient and effective results. GPU-STOMP [51] is combined with a high-performance GPU to allow significantly improving the scalability of motif (discord) discovery. Alaee et al. [32] present the first efficient, scalable and exact method to find time series motifs under DTW. This method automatically performs the best trade-off of time-to-compute versus tightness-of-lower-bounds for a novel hierarchy of lower bounds. He et al. [52] demonstrated the essence of the matrix profile, 1-NN based nonparametric density estimation of subsequences. An ensemble framework, called Neighbor profile [52], was developed to robustly estimate the subsequence density with kNN in a bagging way.

2.2.3 Shapelet-based methods

Shapelets were originally introduced for time series mining with the interpretability in [14]. Then, there have been follow-up studies on shapelets, including logical shapelets [35], shapelet transformation [16], learning shapelets [9], efficient learning shapelets [28] and efficient shapelet discovery [27]. Lines et al. proposed ST [16] for selecting shapelets to represent the original time series. Classification methods such as Neural Network and SVM can be applied to determine shapelets. LTS [9] utilizes logistic regression with stochastic gradient descent to learn shapelets. It increases the classification accuracy of the UCR Time Series Classification Archive. However, the training phase is time-consuming because of the large number of shapelet candidates. A few parameters, such as the numbers and the lengths of shapelets, are manually tuned to select candidates for model building. Grabocka et al. provide fast shapelet discovery method [38] with a distance-based clustering technique for similar shapelet pruning and an incremental NN classifier. Rakthanmanon et al. propose fast shapelets [10], which utilizes a random masking strategy on SAX words of raw subsequences to efficiently discover shapelets. The time complexity is significantly reduced from $O(M^2N^3)$ to $O(MN^2)$ when compared to [14], but the accuracy is often lower than some recent work [2].

2.3 Multivariate Time Series Classification

In this section, we give a brief introduction to the existing methods of multivariate time series classification. We classify them into two main types, namely model-based, and neural network-based.

2.3.1 Model-based Methods

Mei et al. [53] proposed a Mahalanobis distance-based dynamic time warping (DTW) measure, building an accurate relationship between each variate and its

corresponding category for multivariate time series. A tree classifier based on a new symbolic representation to extract information contained in the relationships for multivariate time series is proposed by Baydogan et al. [54]. An accurate and efficient classification method based on common principal components analysis (PCA) to reduce the dimensionality for multivariate time series is proposed in [55]. WEASEL-MUSE [56] is introduced to utilize the bag of SFA (Symbolic Fourier Approximation) to classify multivariate time series. Seto et al. [57] introduced a template selection method based on DTW, which increases the efficiency for feature extracting and bypasses the domain knowledge.

2.3.2 Neural Network-based Methods

Another type is based on neural networks. A nice review paper [22] summarizes many neural networks-based methods for time series classification. LSTM-FCN [58] employs an LSTM layer and stacked CNN layer to extract features for a softmax layer to predict the label for classification. Zheng et al. [59] proposed a deep learning framework, Multi-Channels Deep Convolutional Neural Networks, which learn features from individual univariate time series, then combine all features from each variate into a Multilayer Perceptron (MLP) for classification. [60] applies one positive samples and several negative samples when training their neural network, then SVM is utilized to do the final classification. TapNet [61] is the latest work of this type. It utilizes an attentional prototype network to learn the latent features from multivariate time series. All the methods mentioned above learn an end-to-end classification model, providing little interpretability.

2.4 Time Series Clustering

In this section, we focus on the autoencoder-based and shapelet-based methods on time series clustering. Interested readers may refer to some excellent review papers on time series clustering [4][5].

2.4.1 Autoencoder-based Methods

Deep embedding clustering (DEC) [44] is a general method that simultaneously learns feature representations and cluster assignments for many data-driven application domains using deep neural networks. After learning the lower-dimensional feature space, a clustering objective is optimized iteratively. Guo et al. [45] discovered that the defined clustering loss may corrupt the feature space and cause meaningless representations. The proposed algorithm, improved deep embedding clustering (IDEC), can preserve the structure of the data generating distribution with an under-complete autoencoder. Deep temporal clustering (DTC) [62] naturally integrates an autoencoder network for dimensionality reduction and a novel temporal clustering layer for new time series representation clustering into a single end-to-end learning framework without using labels. DTCR [63] proposes a seq2seq autoencoder representation learning model, integrating reconstruction task (for autoencoder), K-means task (for hidden representation), and classification task (to enhance the ability of encoder). After learning the autoencoder, the clustering method (*e.g.*, K-means) is applied to the hidden representation.

2.4.2 Shapelet-based Methods

The shapelet-based method was introduced with its interpretability in [14], and then studies on logical shapelets [35], shapelet transformation [16], learning shapelets [9], Matrix Profile [29], and efficient learning shapelets [28][27]. The unsupervised-shapelets method (a.k.a U-shapelets) was proposed for time series clustering in [11]. U-shapelets are discovered without label information, then shapelets have been applied on both time series classification and time series clustering. Scalable U-shapelets, a hash-based algorithm to discover U-shapelets efficiently, was introduced by [12]. The optimized techniques of SAX [20] have been exploited in scalable U-shapelets. k-Shape [64] relies on a scalable iterative refinement procedure to

generate homogeneous and well-separated clusters. Zhang et al. [13] proposed an unsupervised salient subsequence learning model for time series clustering, which incorporates shapelet learning, shapelet regularization, spectral analysis and pseudo labeling.

Chapter 3

Efficient Shapelet Discovery for Univariate Time Series Classification - A Heuristic Approach

In this chapter, we investigate the problem of scalable and efficient time series classification. The studied problem and time series classification preliminaries are introduced in Section 3.1. Section 3.2 proposes the details of BSPCOVER for discovering a set of high-quality SAX words for model building. Section 3.3 presents our experimental results. Finally, we summarize this chapter in Section 3.4.

3.1 Preliminaries and Problem Statement

In this section, we further present some specific preliminaries and summarize the notations and their meanings in Table 3.1.

Definition 3.1. PAA sequence $\bar{T}$ [19]. A time series T can be represented by a PAA sequence $\bar{T} = (\bar{t_1}, \bar{t_2}, \cdots, \bar{t_{\lceil \frac{N}{\omega} \rceil}})$, where the i-th element of $\bar{T}$, $\bar{t_i}$, is calculated by the following equation:

$$\bar{t_i} = \frac{1}{\omega} \sum_{j=\lceil (i-1)\omega+1 \rceil}^{\lceil i\omega \rceil} t_j, \tag{3.1}$$

where ω is the number of consecutive values for averaging. □

Hence, ω is sometimes referred as the compression ratio of the PAA sequence

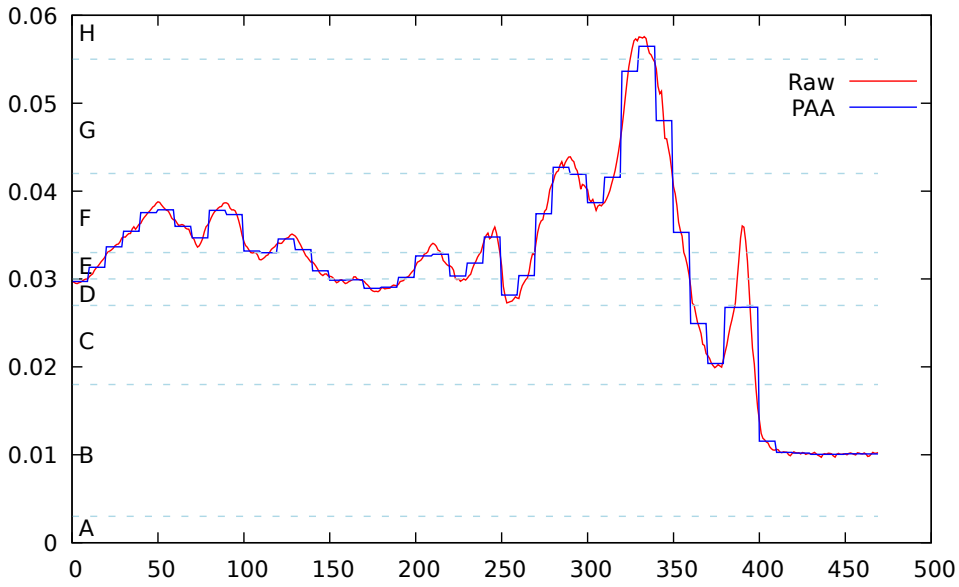


Figure 3.1: An example of PAA with $\omega = 10$ and SAX alphabet size with $|\Sigma| = 8$ from the Beef dataset for SAX sequence **DEFFF**$\cdots$**BBBBB**

over the raw time series data.

Definition 3.2. SAX sequence $\hat{T}$ [20]. A SAX sequence $\hat{T} = \hat{t_1}, \hat{t_2}, \cdots, \hat{t_{\lceil \frac{N}{\omega} \rceil}}$ can be mapped from $\bar{T}$ with an alphabet by the breakpoints of values following the Gaussian distribution [65]: The alphabet $\Sigma = \{A_1, A_2, \cdots, A_d\}$, the size of Σ is the number of characters (a.k.a symbols), denoted as $|\Sigma|$. □

Example 3.1. Figure 3.1 shows a time series data instance (red solid line) from the Beef dataset with the length $n = 470$. The PAA sequence (blue solid line) is with $\omega = 10$, and the SAX sequence is **DEFFF**$\cdots$**BBBBB** (breakpoints are the light blue dash lines) with $|\Sigma| = 8$ characters. □

Putting these together, we are ready to present the problem statement below.

Problem statement. Given an integer k and a time series dataset D, where each time series has a class label, we aim to efficiently discover the shapelets $\mathcal{S} : \{S_1, \cdots, S_{k'}\}$, $k' \leq k$, for each class, such that

$$\arg\max_{\mathcal{S}} \sum_{\substack{T,T' \in D \\ \mathsf{label}(T) \neq \mathsf{label}(T')}} ||\mathsf{dist}(T, \mathcal{S}) - \mathsf{dist}(T', \mathcal{S})||. \tag{3.2}$$

Table 3.1: Summary of frequently used notations in BSPCOVER

Notation	Meaning
$\bar{T}$	the PAA result of time series T
Σ	the alphabet set of SAX words, *e.g.*, {A,B,C,D,E}
$\hat{T}$	the SAX result of time series T
ω	compression ratio of time series data
θ	the threshold for determining ω
ϕ	the minimum sliding window ratio of $\hat{T}$
τ	a small real value

The shapelets discovered are used to learn a classification model. The details are presented in Appendix A.

3.2 Efficient SAX Subsequence Computation

In this section, we present the details of the BSPCOVER approach for discovering a set of SAX subsequences (a.k.a *SAX words*) of high-quality for model building. For presentation clarity, we present BSPCOVER using Algorithm 3.1-3.5.

Overview. The raw time series data are reduced to SAX sequences, then a sliding window is used to generate SAX words of variable lengths (Algorithm 3.1). Mining some high-quality SAX words from voluminous SAX words is inefficient. Our main techniques compute a small set of SAX word candidates as follows. **I.** We construct a bloom filter for each class to efficiently prune the same SAX words that exist in all classes, and build a bitmap structure for each remaining SAX word (Algorithm 3.2); **II.** We prune similar SAX words that appear in all classes, and then define a weighted bitmap of each SAX word (Algorithm 3.3); and **III.** We formulate the discovery problem as a weighted set cover problem, and propose a heuristic method to solve it (Algorithm 3.4 and Algorithm 3.5).

3.2.1 SAX Transformation

For completeness, this subsection recalls the relevant details of the transformation this thesis adopts. BSPCOVER transforms raw time series data into PAA sequences, SAX

sequences, and then SAX words to reduce their information but enhance processing efficiency. The parameters of the transformation can be easily set, since excessive transformed SAX words are further pruned by our techniques.

We present the procedure for compressing each time series T of a dataset D in Algorithm 3.1. It applies PAA to the raw time series data through averaging ω consecutive data values (Lines 5-9). After generating the PAA sequence $\bar{T}$, Algorithm 3.1 transforms $\bar{T}$ to SAX sequence $\hat{T}$ by replacing the PAA sequence value from a lookup table [34] (Line 10). It then generates SAX words through sliding windows (Lines 12-15), whose lengths are determined by the minimum sliding window ratio ϕ (Line 12). We then elaborate upon the two parameters ω and ϕ.

Determining ω. According to the finding in [38], when ω increases from 1 to 5, the shapelet discovery time is significantly reduced and the accuracy remains fairly stable. With ω increasing over 5, the discovery time of shapelets shows no obvious decline but the accuracy deteriorates slightly. Next, according to the experiment results from related studies [35] [10], few shapelets are of the full length of the raw time series. We obtain the same findings from our experiments.

Based on these two findings, to preserve more information of time series, and meanwhile to improve efficiency, we set ω to range from $\omega_{\min} = 1$ to $\omega_{\max} = 5$ for the different datasets, and specify a step function of ω for N. For instance, the step function π used in our experiments for determining ω is shown in Figure 3.2.

Determining ϕ. The minimum sliding window ratio ϕ can be determined for the different datasets as follows. If the lengths of subsequences are smaller than $L_{\min}$, normally 10, their discriminative power is small. Thus, given the minimum length of the subsequence, the minimum sliding window ratio of each dataset can be computed in Line 3 of Algorithm 3.1 and sliding windows of various lengths are generated in

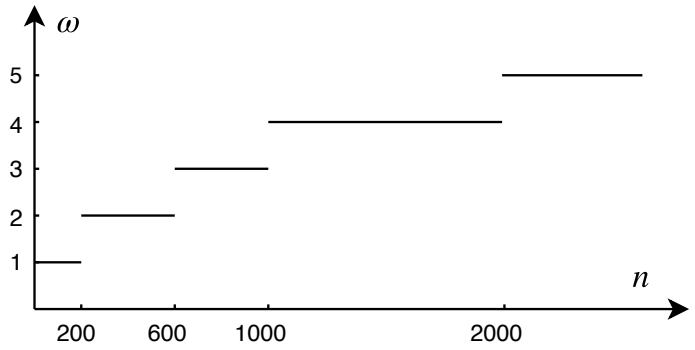


Figure 3.2: The step function π of ω

Line 12. By default, $L_{\min}$ is set to $n \times 10\%$, denoted as $\phi = 10\%$. If the default value of $L_{\min}$ is smaller than 10, we then set $L_{\min}$ to 10 to deduce the ϕ.

Furthermore, it is not necessary to consider subsequences of all possible lengths but those with a difference of ϕ. The rationale is that the discriminative power of subsequence $T(p, q)$ is similar to the subsequence $T(p, q + \Delta)$, if Δ is tiny,

$$\mathsf{dist}(T(p, q), T(p, q + \Delta)) < \tau, \tag{3.3}$$

where τ is a small value.

If the distance between two subsequences is smaller than τ, their discriminative powers are considered similar. Thus, the minimum sliding window ratio is used to generate SAX words (Lines 3 and 12).

Example 3.2. The SAX transformation process, from raw time series to PAA sequences and SAX words, is shown in Figure 3.3. Suppose the instance belongs to a class, with a raw time series length $N = 8$, a compression ratio $\omega = 2$ and an alphabet size $|\Sigma| = 4$, the raw time series is transformed into **CDBD**. The minimum sliding window ratio $\phi = 0.5$; thus, the sliding window sizes are 2 and 4. The SAX words are {**CD, DB, BD, CDBD**} and are appended to the SAX words set Ω_1. □

Complexity. The complexity of Algorithm 3.1 is $O(MN)$, which is composed of two parts. The first PAA sequence part (Lines 6-10) is $O(MN)$; and the time complexity of SAX words generation part (Lines 12-15) is $O(\frac{MN}{\omega\phi})$.

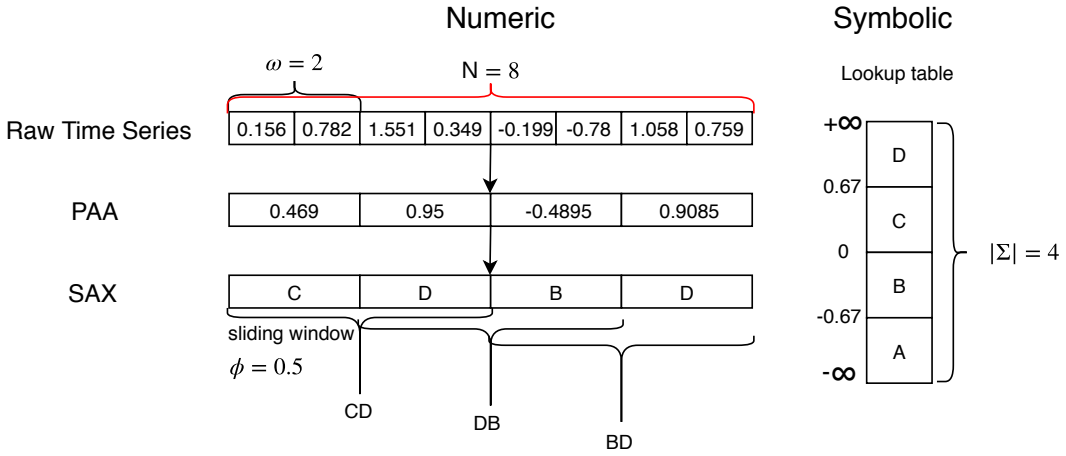


Figure 3.3: An example of SAX transformation

3.2.2 Bloom Filter and Bitmap of SAX Words

Because of the SAX transformation of Section 3.2.1, the number of SAX words generated by Algorithm 3.1 is much larger than those of LTS [9] and ELIS [1]. Only three lengths (at most) of shapelets exist in LTS, which are smaller than one-fifth of ours on average. The maximum length of the compressed times series in ELIS is 50 for the PAA sequences in all datasets, which indicates that the maximum ω of ELIS is over 10. Identical SAX words often exist in different classes, which do not have discriminative power. If these SAX words are selected as candidates for model building, the accuracy does not improve.

We propose to use bloom filters in BSPCOVER to efficiently prune the SAX words that do not have discrimination, *i.e.*, those present in all classes.[1] The bloom filters using 13 independent MD5 hash functions are constructed for each class, to achieve low false positive probability (Lines 3-5, Algorithm 3.2).

3.2.2.1 Filtering non-discriminative SAX words

In the filtering step, each SAX word is treated as the query. For class C, all the SAX words are taken as the query for bloom filters of other classes. There are two types

[1]To the best of our knowledge, this thesis is the first work to use bloom filters to prune SAX words in shapelet discovery. A previous paper [66] utilized bloom filters to store the SAX words for motif discovery.

Algorithm 3.1: SAX transformation for class C

Input: Time series dataset $D = T^{M \times N}$
Output: SAX words set Ω_C for each class C

Initialize $\Omega_C = \varnothing$;
$\omega = \pi(N)$; // Step function shown in Figure 3.2
$\phi = \frac{L_{\min}}{\lceil \frac{N}{\omega} \rceil}$;
foreach $j \in \{1, 2, \cdots, M\}$ **do**
    {PAA and SAX sequence generation}
    foreach $i \in \{1, 2, \cdots, \lceil \frac{N}{\omega} \rceil\}$ **do**
        foreach $r \in \{\lceil (i-1)\omega + 1 \rceil, \cdots, \lceil i\omega \rceil\}$ **do**
            $\bar{T}^{j \times i} = \bar{T}^{j \times i} + T^{j,r}$;
        end
        $\bar{T}^{j \times i} = \frac{\bar{T}^{j \times i}}{\omega}$;
        $\hat{T}^{j \times i} =$ SAX lookup table$(D, \bar{T}^{j \times i})$ [34];
    end
    {SAX words generation}
    foreach $l \in \{\phi \cdot \lceil \frac{N}{\omega} \rceil, 2 \cdot \phi \cdot \lceil \frac{N}{\omega} \rceil, \cdots, \lceil \frac{N}{\omega} \rceil\}$ **do**
        foreach $x \in \{1, 2, \cdots, \lceil \frac{N}{\omega} \rceil - l + 1\}$ **do**
            $\hat{e} = \hat{T}(x, x + l - 1)$;
            $\Omega_C = \Omega_C + \hat{e}$;
        end
    end
end
return Ω_C

of query results. The first is "*possibly in set*", which means the SAX word exists in other classes with high probability and is removed in all classes' candidates. The other is "*definitely not in set*", which indicates that this SAX word is specifically found in its own class, and therefore, can only represent class C. Thus, it is kept in $\Omega_C^{\mathcal{BF}}$ for further processing in Section 3.2.3. The process of pruning is shown in Lines 8-12 in Algorithm 3.2.

Example 3.3. The function of bloom filters is illustrated with Figure 3.4. We generate a bloom filter for each class (shown in ①) and take one SAX word $\hat{e}$ from Ω_1 as the query for the other two bloom filters generated by Ω_2 and Ω_3 in ② and ④. Two kinds of operations are shown in ⑥. If the SAX word can pass the bloom filters, this SAX word $\hat{e}$ may have high discriminative power and can be inserted into the

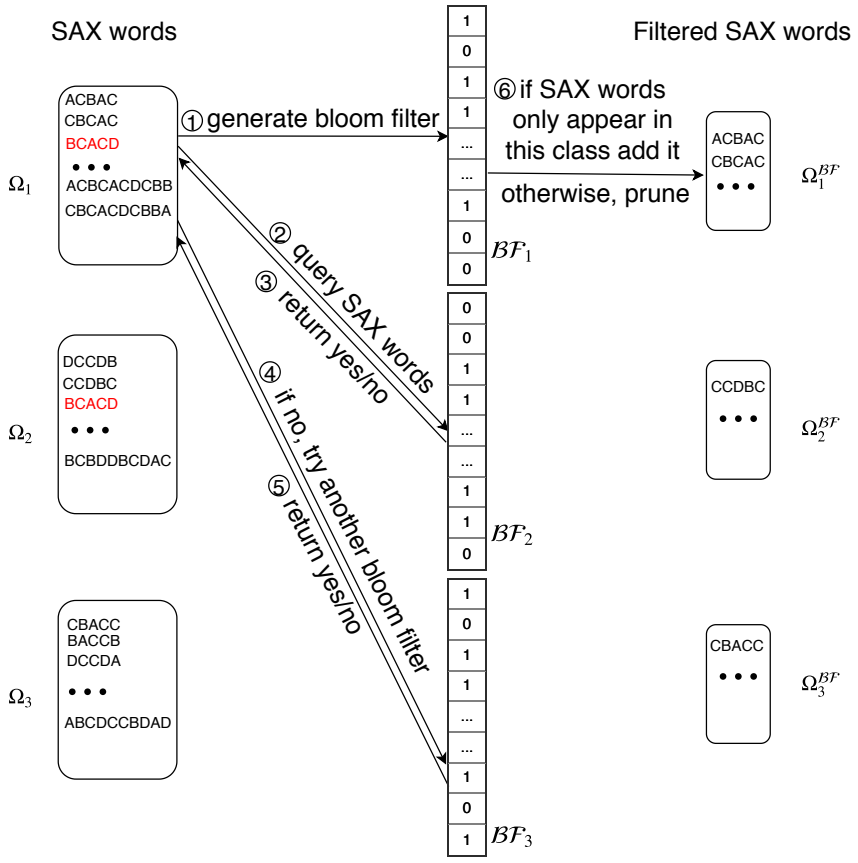


Figure 3.4: Pruning redundant and non-discriminative SAX words (of 3 classes) using bloom filters

candidates set $\Omega_1^{\mathcal{BF}}$. Otherwise, the SAX word is pruned, such as **BCACD**, which exists in Ω_1 and Ω_2. Then, we continue the next SAX word until all words in Ω_1 are processed and carry out the same operation for SAX words of Ω_2 and Ω_3. □

3.2.2.2 Building compact representation of SAX words

To quantify the quality of candidate SAX words $\Omega_C^{\mathcal{BF}}$ of a class C, we propose to generate a bitmap structure for each SAX word $\hat{e}$ (Lines 15-20 in Algorithm 3.2). The length of bitmap is the number of instances in the class $|D_C|$ and the initial value of each bit is set to 0. If $\hat{e}$ exists in the sequence $\hat{T}_j$, then $\hat{e}.\mathsf{bitmap}(j) = 1$ and we say that $\hat{e}$ *covers* $\hat{T}_j$. That is, the value of j-th position of bitmap indicates whether the SAX word candidate covers $\hat{T}_j$. The larger the number of 1 values of the bitmap of $\hat{e}$, the more instances $\hat{e}$ covers and the better the quality of $\hat{e}$.

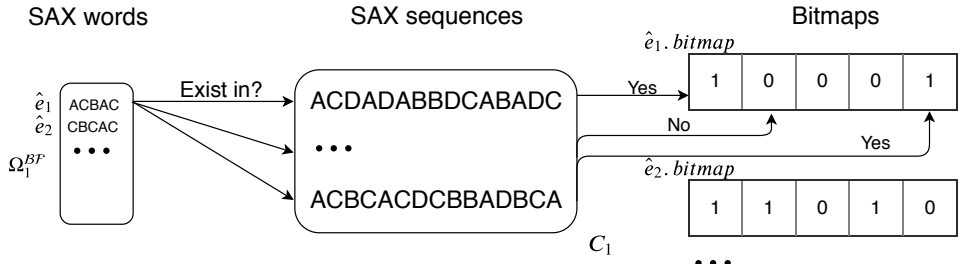


Figure 3.5: An example of (partial) constructing bitmaps of SAX words in Class 1 (C_1)

Algorithm 3.2 computes the bitmaps of all SAX words in Lines 15-20. In the end, we obtain the candidates set and their bitmaps. Figure 3.5 shows an example of (partial) bitmaps for $\Omega_1^{\mathcal{BF}}$.

Complexity. The complexity of Algorithm 3.2 is $O(\frac{MN}{\omega\phi}|D_C|)$, which is composed of three parts: Lines 3-5 of $O(\frac{MN}{\omega\phi})$, Lines 8-12 of $O(\frac{MN}{\omega\phi})$ and Lines 15-20 of $O(\frac{MN}{\omega\phi}|D_C|)$.

3.2.3 Similar SAX Word Pruning

In the previous subsection, the bloom filter is proposed to prune duplicated SAX words that have low discriminative power. There are still often numerous similar SAX words in different classes. For example, after the pruning, over 50% of all SAX words of the dataset **Beef** are similar. Such similar words in different classes' shapelet candidates evidently reduce both efficiency and accuracy.

A natural technique for reducing excessive SAX words for model building is to prune similar SAX words that exist in all classes. There are two technical details. First, similarity measures of time series violate triangle inequality. That is, if T_1 is similar to T_2, T_2 is similar to T_3, and T_1 is not necessarily similar to T_3. Second, it is required to differentiate the similar SAX words that exist in self class and other classes and to select one SAX word to represent similar ones in self class. In this subsection, we define a non-metric distance measure for the similarity measure of SAX words. As for the second technical issue, we propose to use term frequency as

Algorithm 3.2: Construction of bloom filters and the SAX words' bitmaps of class C

Input: SAX words set Ω_C

Output: Candidate set with their bitmaps stored in bloom filter $\Omega_C^{\mathcal{BF}}$

{Constructing bloom filters}
Initialize $\mathcal{BF}_C$ for each C in $\mathcal{C}$;
foreach $C \in \mathcal{C}$ **do**
    foreach $\hat{e} \in \Omega_C$ **do**
        $\mathcal{BF}_C$.add($\hat{e}$) ; // using MD5 hash functions
    end
end

{Pruning duplicate SAX words}
foreach $C \in \mathcal{C}$ **do**
    Initialize $\Omega_C^{\mathcal{BF}} = \varnothing$;
    foreach $\hat{e} \in \Omega_C$ **do**
        if $\bigvee_{\bar{C}=\mathcal{C}\setminus\{C\}} \mathcal{BF}_{\bar{C}}$.lookup($\hat{e}$) == false **then**
            $\Omega_C^{\mathcal{BF}}$.add($\hat{e}$) ;
        end
    end
end

{Constructing bitmaps}
foreach $C \in \mathcal{C}$ **do**
    foreach $\hat{e} \in \Omega_C^{\mathcal{BF}}$ **do**
        $\hat{e}$.bitmap $= 0^{|D_C|}$;
        for $j \in \{1, 2, \cdots, |D_C|\}$ **do**
            if $\hat{e} \in \hat{T}_j$ **then**
                $\hat{e}$.bitmap(j) $= 1$;
            end
        end
    end
end
return $\Omega_C^{\mathcal{BF}}$

the weight of a SAX word to quantify the quality of SAX words. The greater the weight, the better the quality.

In Def. 3.4, we propose a non-metric distance for SAX words. When we use it to determine similar SAX words (Def. 3.5), we propose a threshold, which is dependent on the words' lengths (specifically, $\frac{l}{\phi \cdot \lceil \frac{N}{\omega} \rceil}$).

Definition 3.3. SAX L1 distance (L1D)**.** Given two SAX words with the same length, $\hat{e}_1 = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_l)$ and $\hat{e}_2 = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_l)$, the *SAX L1 distance between* $\hat{e}_1$ *and* $\hat{e}_2$ is defined as follows.

$$\text{L1D}(\hat{e}_1, \hat{e}_2) = \Sigma_{i=1}^{l} |\hat{x}_i - \hat{y}_i| \tag{3.4}$$

□

Definition 3.4. Matching distance of SAX words (MATCHD)**.** Given two SAX words of the same length, $\hat{e}_1 = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_l)$ and $\hat{e}_2 = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_l)$, four kinds of subsequences are defined as $\hat{e}_1^{1+i} = (\hat{x}_{1+i}, \hat{x}_{2+i}, \cdots, \hat{x}_l)$, $\hat{e}_2^{l-i} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{l-i})$, $\hat{e}_1^{l-i} = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_{l-i})$, and $\hat{e}_2^{1+i} = (\hat{y}_{1+i}, \hat{y}_{2+i}, \cdots, \hat{y}_l)$, where $i \in [1, \frac{l}{\phi \cdot \lceil \frac{n}{\omega} \rceil}]$.

The *matching distance of SAX words* $\hat{e}_1$ *and* $\hat{e}_2$, denoted as $\text{MATCHD}(\hat{e}_1, \hat{e}_2)$, is defined below.

$$\text{MATCHD}(\hat{e}_1, \hat{e}_2) = \min\{\text{L1D}(\hat{e}_1^{1+i}, \hat{e}_2^{l-i}), \text{L1D}(\hat{e}_2^{1+i}, \hat{e}_1^{l-i})\}, \tag{3.5}$$

□

Some SAX words do not match well may be just because they are generated by sliding window and they do not align well. Hence, we generate $\hat{e}_1^{1+i}$ and $\hat{e}_1^{l-i}$ by removing i characters ($i \in [1, \frac{l}{\phi \cdot \lceil \frac{N}{\omega} \rceil}]$) from the beginning and the end of $\hat{e}_1$. We generate $\hat{e}_2^{1+i}$ and $\hat{e}_2^{l-i}$ similarly. MATCHD uses them to compute the smallest distance between the matching of the middle parts of $\hat{e}_1$ and $\hat{e}_2$. Such definition leads to effective pruning of similar SAX words.

Example 3.4. Consider the dataset Beef and two of its SAX words $\hat{e}_1$ = **BCCDA** and $\hat{e}_2$ = **CCDAA**. **I.** For the L1 distance, L1D($\hat{e}_1, \hat{e}_2$) = 1+0+1+3+0 = 5; **II.** For MATCHD, the maximum removing number of these two SAX words is $\frac{5}{\lceil 0.05 \times \frac{470}{5} \rceil} = 1$, MATCHD between them is the minimum of two following situations, one is removing the first character of $\hat{e}_1$ and the last character of $\hat{e}_2$ L1D(**BCCD**, **CCDA**) $= 5$; another is L1D(**CCDA**, **CCDA**) $= 0$. Hence, MATCHD($\hat{e}_1, \hat{e}_2$) = 0. □

Definition 3.5. Similar SAX words. Given two SAX words of the same length, $\hat{e}_1 = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_l)$ and $\hat{e}_2 = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_l)$, ISSIMILAR is defined as follows: ISSIMILAR($\hat{e}_1, \hat{e}_2$) =

$$\begin{cases} \text{true}, & \text{MATCHD}(\hat{e}_1, \hat{e}_2) = 0, \text{or} \\ & \text{L1D}(\hat{e}_1, \hat{e}_2) \leq \frac{l}{\phi \cdot \lceil \frac{N}{\omega} \rceil} \text{ and } \bigwedge_{i \in [1,l]} |\hat{x}_i - \hat{y}_i| \leq 1, \\ \text{false}, & \text{otherwise.} \end{cases} \tag{3.6}$$

□

Formula 3.6 states that $\hat{e}_1$ and $\hat{e}_2$ are similar if there is a match using MATCHD. Alternatively, two SAX words are similar if they are of the same length l and the distance (Def. 3.3) is no larger than a ratio between l and the minimum sliding window size $\phi \cdot \lceil \frac{N}{\omega} \rceil$. Since the SAX words are generated by $\phi \cdot \lceil \frac{N}{\omega} \rceil$, l can be divisible by the minimum sliding window size (Line 3, Algorithm 3.1).

The pseudocode of pruning similar SAX words and counting the weights of the remaining SAX words are shown in Algorithm 3.3. If a SAX word $\hat{e}_1$ has similar SAX words in another class (Line 5), Algorithm 3.3 prunes those similar words, according to Eqn. 3.6 (Lines 6-7). $\hat{e}_1$ is also pruned (Line 10) because it is not discriminative.

Next, in Lines 12-21, we compute the weight of each SAX word, which records the number of similar SAX words in its class. Algorithm 3.3 records each weighted SAX word in the set Ω_C^{sim}. Ω_C^{sim} is initialized to an empty set (Line 14). Algorithm 3.3 calculates the similarity with Eqn. 3.6 between each SAX word and others in $\Omega_C^{\mathcal{BF}}$

(Lines 18-19) and updates the weight (Lines 20-21).

Complexity. The complexities of Lines 1-10 and Lines 12-21 are both $O(M(\frac{N}{\omega\phi})^2)$. Hence, the time complexity of Algorithm 3.3 is $O(M(\frac{N}{\omega\phi})^2)$.

Algorithm 3.3: Computing the weights for non-similar SAX words

```
1  Input: Candidates set with bloom filter Ω^{BF}
   Output: Weighted candidates after pruning, Ω^{sim}
2  {Pruning similar words}
3  foreach C ∈ C do
4      foreach ê_1 ∈ Ω_C^{BF} do
5          foreach ê_2 ∈ Ω_{C\{C}}^{BF} do
6              if isSimilar (ê_1, ê_2) then
7                  Ω_{C\{C}}^{BF}.remove(ê_2) ;
8                  flag = true;
9              end
10         end
11         if flag then
12             Ω_C^{BF}.remove(ê_1) ;
13         end
14     end
15 end
16
17 {Counting the weights of remaining SAX words}
18 foreach C ∈ C do
19     Initialize Ω_C^{sim} = ∅ ;
20     while Ω_C^{BF} ≠ ∅ do
21         ê_1 = Ω_C^{BF}.pop() ;
22         Ω_C^{sim}.add(ê_1) ;
23         foreach ê_2 ∈ Ω_C^{BF} \ {ê_1} do
24             if isSimilar(ê_1, ê_2) then
25                 ê_1.weight++ ;
26                 ê_2.weight++ ;
27             end
28         end
29     end
30 end
31 return Ω^{sim}
```

3.2.4 SAX Word Cover for Model Building

In this subsection, the details of the BSPCOVER method to compute SAX words for model building are presented. In a nutshell, the problem is formulated as a weighted bitmap cover problem, through finding the minimum subset to cover the time series instances rather than discovering the SAX words directly. We provide some definitions for explaining the approach below.

Definition 3.6. p**-Cover.** A SAX word $\hat{e}$ *covers* a SAX sequence $\hat{T}_j$ *if and only if* there is a subsequence of $\hat{T}_j$ that is identical to $\hat{e}$. SAX sequence $\hat{T}_j$ is p-Covered *if and only if* $\hat{T}_j$ is covered by p distinct SAX words. □

To implement the covering relationships between SAX words and SAX sequences of class C, we utilize the bitmap structure. The number of bits of bitmap is $|D_C|$, and $\hat{e}$.bitmap $(j) = 1$ if and only if $\hat{e}$ covers $\hat{T}_j \in D_C$.

Definition 3.7. Weighted bitmap cover (WBC) **problem.** Given a dataset D_C and a set of SAX words $\Omega_C^{sim} = \{\hat{e}_1, \cdots, \hat{e}_q\}$, the *weighted bitmap cover* problem is to determine a subset Ω^{fin} of Ω_C^{sim} such that:

- all $\hat{T}_j$ are p-covered by Ω^{fin}, and
- the total weight of Ω^{fin} is maximized. □

When $p = 1$, the weighted bitmap cover problem is equivalent to the weighted set cover problem, which is an NP-complete problem. The p value of the WBC problem is not smaller than 1 and hence, WBC is also an NP-complete problem.

It is known that the weighted set cover problem can be practically solved by Chvatal [67], which follows one rule: choosing the set that contains the largest number of uncovered elements at each stage. The approximation ratio is $\log q$, where q is the number of SAX subsequences in Ω_C^{sim}. Therefore, we extend the greedy algorithm to Algorithm 3.4 to solve the WBC.

Algorithm 3.4 finds candidate SAX words for each class C. It initializes the final SAX word set Ω_C^{fin} and a special bitmap, called Bitmap_C. Its length is $|D_C|$ and all of its bits have a value of 0 (Lines 3-4), where $\mathsf{Bitmap}[i] = 1$ when the instance has been covered by some of the SAX words in Ω_C^{fin}, or 0 otherwise. Next, Algorithm 3.4 sorts the SAX words based on the time series they cover and then the number of similar SAX words (*i.e.*, the number of 1 values and weight of their bitmaps) in descending order (Line 5). A larger number of 1 values and a greater weight indicate that the SAX word is of a higher quality.

Algorithm 3.4 selects the next SAX words from Ω_C^{sim}, adds it into Ω_C^{fin} (Lines 8-9), then adds its bitmap to Bitmap (Line 10). Then, if Bitmap = 1, all instances of the class are covered by the current SAX words, and Algorithm 3.4 breaks the loop and finds SAX words for the next class (Lines 11-12). Otherwise, Bitmap is not equal to 1 and the similar SAX words of $\hat{e}_1$ in Ω_C^{sim} are removed so that they are not passed to the potentially costly model building phase (Lines 15-17). The remaining candidates are sorted, similar to Line 5, at the positions that are not covered by Ω_C^{sim} (Lines 18-19). Then, Algorithm 3.4 processes the remaining SAX words iteratively, from Line 20.

Analysis. The complexity of Algorithm 3.4 is $O(M(\frac{N}{\omega\phi})^2)$. The approximation ratio of Algorithm 3.4 is $\log q$.

Example 3.5. Figure 3.6 shows the *whole* process of finding SAX words from raw time series. All the time series instances are transformed into SAX words, as shown in ①. Then, the SAX words are utilized to build the bloom filter $\mathcal{BF}_C$ for each class C, which prunes identical SAX words in other classes (blue dotted rectangle). Next, the bitmap structures for the remaining SAX words are built to quantify their qualities (red dotted rectangle). ② The procedure prunes similar SAX words in all classes. The red SAX words **DCCCC** and **DCCCD** of different classes are similar, whereas the blue SAX words of the same class are similar. The red ones are pruned and the

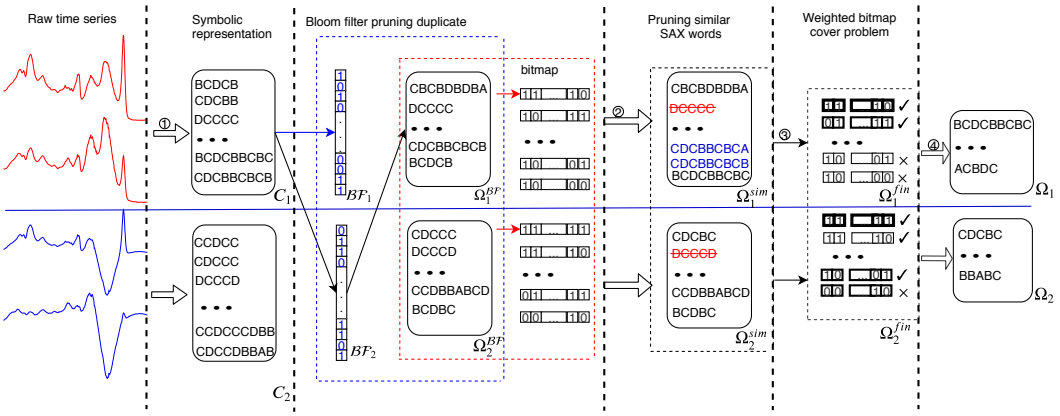


Figure 3.6: An example of finding SAX words for model building (The figure is best viewed in color)

blue ones are passed to compute their weights. ③ Algorithm 3.4 utilizes the weight bitmap structures (the last rectangle) to select the minimum subset of bitmaps to cover all the instances. □

Algorithm 3.5 (Lines 2-7) applies Algorithm 3.4 p times to determine the p-cover SAX words. The approximation ratio is $(\log q)^p$.

It should be noted that the SAX words Ω_C^{fin} after the loop (Lines 3-7) may cover some time series instances more than p times. A large Ω_C^{fin} reduces the efficiency of model building. Hence, if Ω_C^{fin} has more than k shapelets, we greedily prune some SAX words (Lines 13-19 of Algorithm 3.5). This is again implemented by a bitmap Bitmap_C. $\mathsf{Bitmap}_C[j]$ records the number of times T_j is covered by Ω_C^{fin} (Lines 11-12). If there is a T_j covered more than p times and the number of shapelets is still larger than k (Line 13), we implement a function mincost (Line 14) to compute an $\hat{e}$ has the smallest cost and after the removal of $\hat{e}$, Ω^{fin} is still a p-cover. The cost of $\hat{e}$ is the number of time series $\hat{e}$ covers. If there is such an $\hat{e}$, we remove it from Ω_C^{fin} (Lines 18-19).

3.2.5 p-Cover Setting

Determining p. The pseudocode of Algorithm 3.6 shows how to determine p. In

Algorithm 3.4: OneSaxCover: Heuristic algorithm for the weighted bitmap cover problem

Input: Candidate set after pruning Ω^{sim}
Output: Final candidate set Ω_C^{fin}, for all $C \in \mathcal{C}$
foreach $C \in \mathcal{C}$ **do**
  Initialize $\Omega_C^{fin} = \varnothing$;
  $\mathsf{Bitmap}_C = 0^{|D_C|}$; // all pos. are uncovered
  $\Omega_C^{sim}.\mathsf{sort}()$; // by the number of 1 values and then the weight of bitmap of SAX words in Ω_C^{sim}
  {Select k SAX words}
  do
    $\hat{e}_1 = \Omega_C^{sim}.\mathsf{pop}()$;
    $\Omega_C^{fin}.\mathsf{add}(\hat{e}_1)$;
    $\mathsf{Bitmap}_C = \mathsf{Bitmap}_C + \hat{e}_1.\mathsf{bitmap}$; // mark covered pos. by $\hat{e}_1$
    if $\mathsf{Bitmap}_C == 1^{|D_C|}$ **then**
      break ; // fully covered
    end
    else
      {Pruning similar SAX words}
      foreach $\hat{e}_2 \in \Omega_C^{sim} \setminus \{\hat{e}_1\}$ **do**
        if isSimilar$(\hat{e}_1, \hat{e}_2)$ **then**
          $\Omega_C^{sim}.\mathsf{remove}(\hat{e}_2)$;
        end
      end
      $\mathsf{Pos} = \mathsf{Bitmap}_C.\mathsf{pos}()$; // uncovered pos.
      $\Omega_C^{sim}.\mathsf{sort}(\mathsf{Pos})$; // by the number of 1 values from uncovered pos., then the weight of bitmap of SAX words in Ω_C^{sim}
    end
  while $\mathsf{Bitmap}_C \neq 1$ // pos. are not fully covered;
end
return Ω_C^{fin} (for all $C \in \mathcal{C}$)

Algorithm 3.5: PSAXCOVER: Determining p-Cover SAX words

Input: Ω^{sim}, p, k
Output: Final candidates Ω^{fin}
Initialize $\Omega^{fin} = \varnothing$ and $i = 0$;
while $i < p$ **do**
    $\Omega^{fin'}$ = ONESAXCOVER(Ω^{sim}) ;
    $\Omega^{fin} = \Omega^{fin} + \Omega^{fin'}$;
    $\Omega^{sim} = \Omega^{sim} - \Omega^{fin'}$;
    i++ ;
end
{Pruning Ω_C^{fin} to avoid overly-covered time series}
if $|\Omega_C^{fin}| > k$ **then**
    foreach $C \in \mathcal{C}$ **do**
        Initialize $\mathsf{Bitmap}_C = 0^{|D_C|}$;
        foreach $\hat{e} \in \Omega_C^{fin}$ **do**
            $\mathsf{Bitmap}_C = \mathsf{Bitmap}_C + \hat{e}.\mathsf{bitmap}$;
        end
        while $\exists\ j$ s.t. $\mathsf{Bitmap}_C[j] > p$ and $|\Omega_C^{fin}| > k$ **do**
            $\hat{e} = \Omega_C^{fin}.\mathsf{mincost}()$;
            if $\hat{e}$ is null **then**
                break ;
            end
            else
                $\Omega_C^{fin}.\mathsf{remove}(\hat{e})$;
                $\mathsf{Bitmap}_C = \mathsf{Bitmap}_C - \hat{e}.\mathsf{bitmap}$;
            end
        end
    end
end
return Ω^{fin}

Lines 3 and 6, BSPCOVER denotes the composition of Algorithm 3.1-3.5. First, we call BSPCOVER using $p = 1$ and then learn a model to compute its accuracy. In the do-while loop, Algorithm 3.6 computes the model at $p + 1$ (Lines 6-7), and the ratio between accuracy and time (Line 8). If the ratio is smaller than the user-specified threshold (Δ), p is returned (Lines 9-10, 15). Otherwise, the current accuracy of p is updated by that of $p + 1$, the value of p is updated and the loop continues (Lines 11-14).

Algorithm 3.6: Determining p for BSPCOVER

```
1  Input: Accuracy threshold Δ, and time series dataset D, number of
         shapelets k
   Output: p
2  Initialize p = 1 ;
3  Ω^fin = BSPCOVER(D, p, k) ;
4  acc_p = SHAPELETSLEARNING(Ω^fin) ;                    // LTS [9]
5  do
6      Ω^fin' = BSPCOVER(D, p + 1, k) ;
7      acc_{p+1} = SHAPELETSLEARNING(Ω^fin') ;
8      Δ_cur = (acc_{p+1} − acc_p) / (T_{p+1} − T_p) ;
9      if Δ_cur > Δ then
10         break ;
11     end
12     acc_p = acc_{p+1} ;
13     p = p + 1 ;
14     Ω^fin = Ω^fin' ;
15 while true;
16 return p
```

Cost model for setting parameters. We then present a cost model for BSPCOVER to analyze its efficiency for a dataset D. The efficiency of BSPCOVER is mainly contributed by three components: **I.** the number of iterations I; **II.** the number of cover times p; and **III.** other factors ϵ, such as program initialization and operating system. From the experiments, we observe that the relationships between time and

the factors are linear. Thus, the total time cost can be estimated as follows.

$$T_{\text{BSPCOVER}} = T_I + T_p + \epsilon = c_1 I + c_2 p + \epsilon, \tag{3.7}$$

where both I and $p \in \mathbb{N}$. T_I represents the time of iterations; T_p denotes the time caused by the covering time p; and $c_{i,i\in 1,2}$ are the constants scaling factors for I and p.

Remarks. The SAX words discovered by BSPCOVER (Ω^{fin}) are passed to a model learning algorithm. This thesis has revised [9]. For completeness, we summarize the algorithm in Appendix A. We remark that BSPCOVER can be incorporated with other model learning algorithms.

3.3 Experimental Results

This section presents an experimental evaluation of BSPCOVER applied to the well-known benchmark datasets used for time series classification research. The major findings are that BSPCOVER is both efficient and accurate, more so than two methods used as benchmark classifiers, Rotation Forest [2] and 1NN-DTW [68], and many other shapelet-based methods including ST [16], LTS [9], ELIS [1], Fast shapelets [10], Scalable Shapelet Discovery [38] and COTE [21]. In addition, we present some performance characteristics of BSPCOVER by varying some important parameters. The source code used for this experiment is publicly available at `https://goo.gl/sN9Kz7`. For sustainability, we put the source code on google drive [2].

3.3.1 Environment

We implemented the proposed algorithms in JAVA. The model building (shapelet learning) part was implemented using multithreading, whereas other parts of our algorithms were single-threaded. All the experiments were conducted on a machine

[2] `https://drive.google.com/file/d/1xl5CLzyGSwOHMkZgqZ64jBU1CSVbKY8u/view?usp=sharing`

with a Xeon E5-2630 v4 @ 2.2GHz (2S/10C) / 256GB RAM / 128GB SWAP, running on CentOS 7.6 (64-bit). The parameters of the bloom filters of each dataset can be easily set because they are memory efficient.

3.3.2 Datasets and Parameters

A well-known benchmark of time series classification datasets, namely UCRArchive [17], was tested. We selected 45 representative datasets from the data type *Device (4), ECG (5), Image (11), Motion (7), Sensor (8), Stimulated (5), and Spectro (5)*; the number in parentheses is the number of datasets of the type (about half of each data type). Table 3.2 shows the settings of the experiments with the datasets, where *Train, Test, Class,* and *Length* are the numbers of time series in the training set and the testing set, the number of classes, and the length of time series, respectively.

By running experiments and referencing previous work [1] [9], we tuned several important parameters of our method, including the learning rate η, the regularization λ, the number of iterations I in shapelet learning and the optimal p of the WBC algorithm. The default value of η was set to a small value 0.1 in consideration of the performance and the chances of reaching local minima. λ was chosen from $\{0.01, 0.1, 1\}$, while I was selected from $\{1000, 2000, 3000, 5000\}$. The range of p was from $[1, 6]$. The values of the parameters of each dataset are shown in Table 3.2.

3.3.3 Baselines

We compared BSPCOVER with ten different methods. We provide brief details of each method. Interested readers may refer to the original paper for details.

- **Two benchmarks [2].** Two benchmark classifiers (**RotF** and **DTW-Rn-1NN**) are more competitive than many methods. Hence, we include them in our experiments.
- **Shapelet Transformation (ST) [16].** This method combines a weighted ensemble of standard classifiers, and new time series are classified with a

Table 3.2: Datasets and parameters on the tested UCR ARCHIVE

Dataset	Train	Test	Class	Length	λ	I	p
ArrowHead	36	175	3	251	0.01	1000	5
Beef	30	30	5	470	0.01	1000	5
Beetle/Fly	20	20	2	512	0.01	1000	6
CBF	30	900	3	128	0.01	1000	4
ChlorineConcentration	467	3840	3	166	0.01	3000	3
Coffee	28	28	2	286	0.01	1000	2
Computers	250	250	2	720	0.1	1000	3
CricketZ	390	390	12	300	0.1	1000	5
DiatomSizeReduction	16	306	4	345	0.01	3000	5
DistalPhalanxOutlineCorrect	276	600	2	80	0.01	1000	1
Earthquakes	139	322	2	512	0.1	1000	1
ECG200	100	100	2	96	0.1	1000	2
ECG5000	500	4500	5	140	0.1	1000	1
ECGFiveDays	23	861	2	136	0.01	1000	1
ElectricDevices	8926	7711	7	96	0.01	1000	1
FaceAll	560	1690	14	131	0.01	1000	3
FaceFour	24	88	4	350	1	1000	4
FacesUCR	200	205	14	131	1	3000	6
FordA	1320	3601	2	500	0.1	1000	1
GunPoint	50	150	2	150	0.1	1000	3
Ham	109	105	2	431	0.1	1000	1
HandOutlines	370	1000	2	2709	0.01	1000	1
Haptics	155	308	5	1092	0.1	1000	2
InlineSkate	100	550	7	1882	0.01	1000	6
InsectWingbeatSound	220	1980	11	256	0.01	1000	2
LargeKitchenAppliances	375	375	3	720	0.1	1000	2
Mallat	55	2345	8	1024	0.01	1000	3
Meat	60	60	3	448	0.1	1000	2
NonInvasiveFetalECGThorax1	1800	1965	42	750	0.1	1000	1
OSULeaf	200	242	6	427	0.1	3000	4
Phoneme	214	1896	39	1024	0.01	1000	1
RefrigerationDevices	375	375	3	720	0.01	1000	1
ShapeletSim	20	180	2	500	0.1	5000	6
SonyAIBORobotSurface1	20	601	2	70	0.01	5000	5
SonyAIBORobotSurface2	27	953	2	65	0.01	1000	3
Strawberry	370	613	2	235	0.1	1000	4
Symbols	25	995	6	398	0.1	1000	4
SyntheticControl	300	300	6	60	0.1	1000	2
ToeSegmentation1	40	228	2	277	0.01	1000	1
TwoLeadECG	23	1139	2	82	0.1	5000	5
TwoPatterns	1000	4000	4	128	0.01	2000	3
UWaveGestureLibraryY	896	3582	8	315	0.01	1000	2
Wafer	1000	6164	2	152	0.1	1000	4
WormsTwoClass	77	181	2	900	0.1	1000	4
Yoga	300	3000	2	426	0.01	5000	5

weighted vote.

- **Learning Time-Series Shapelets (LTS) [9].** Logistic regression with gradient descent adjusts the selected shapelets for TSC. It has significantly improved accuracy.
- **Fast Shapelets (FS) [10].** FS combines SAX and random masking techniques to enhance efficiency.
- **Scalable Shapelet Discovery (SD) [38].** SD filters out candidates improving classification with an online clustering/pruning technique. The dimensionality reduction ratio r is set to $1/2$ and pruning distance percentile p is 25 for simplicity if there is no support from [38].
- **ELIS [1].** ELIS is the current state-of-the-art of efficient shapelet-based methods. It employs PAA and TF-IDF to improve the efficiency of discovering shapelet candidates. Then, the logistic regression classifier is applied to adjust the shapelets. We follow ELIS [1] to set parameters whenever possible. Otherwise, the parameters are fixed as follows: the iteration number is 1000; the regularization λ is 0.01; and the learning rate η is set to 0.1.
- **COTE [21].** COTE is the meta ensemble method, which is a combination of classifiers.
- **ResNet [22].** The main characteristic of ResNet is the shortcut residual connection between consecutive convolutional layers.
- BSPCOVER**-Random**. (*abbrev.* **Random**) The only difference between this algorithm and BSPCOVER is that it randomly selects the same number of final SAX word candidates as BSPCOVER does. That is, **Random** replaces our heuristic algorithm (in Sec. 3.2.4) with random selection.

3.3.4 Experiments on Efficiency

We observe from our results that the efficiency of ELIS is about two orders of magnitude faster than that of LTS [1]. On the other hand, we find that COTE and

ST are slower than LTS [2]. Thus, we compare FS and ELIS with BSPCOVER to investigate efficiency.

3.3.4.1 Comparison with ELIS and FS, SD

We present the runtime of the total time of ELIS, FS, SD and our method BSPCOVER, and the improvement of FS and SD over BSPCOVER and that of BSPCOVER over ELIS, in Table 3.3. (The unit of the numbers is seconds by default, and we use d and h to denote for days and hours, respectively.)

On these 45 datasets, BSPCOVER is consistently faster than ELIS: 70 times (on average) faster in terms of total time. In particular, it is the fastest method on $\mathbf{MoteStrain}$, whose efficiency is 315 times superior to ELIS. One reason is that our method finds the discriminative shapelets more efficiently prior to model building. This can be verified with the shapelet discovery time, which is 26 times faster than ELIS on average. The iteration number of our method is smaller, enhancing efficiency 80 times over in model building. It confirms the shapelet candidates discovered by our method of high quality. Compared to FS and SD, BSPCOVER is slower in most datasets. While FS and SD are 13 and 488 times faster than BSPCOVER on average, it is known to result in a *significant* drop in accuracy. Our results also reveal that BSPCOVER is clearly more accurate than FS and SD, which exhibit the worst accuracies among all compared methods.

3.3.4.2 Efficiency by varying p

Figure 3.7 presents the total time with the increase of p. For the first three datasets $\mathbf{Beetle/Fly}, \mathbf{Diatom.}, \mathbf{FaceFour}$, the times maintain an approximate linear relationship with p. Regarding $\mathbf{TwoLeadECG}$, there is a similar relationship when p is small. However, the total time stabilizes when p exceeds 5. The reason is that all the candidates for representing the classes have been discovered when

Table 3.3: Efficiency of BSPCOVER and related methods on UCR ARCHIVE (units is second; d and h denotes days and hours)

Dataset	SD	FS	ELIS	BSPCOVER	Speedup SD vs BSP-COVER	Speedup FS vs BSP-COVER	Speedup BSP-COVER vs ELIS
ArrowHead	2.83	7.68	3.4h	55.57	19.63	7.23	220.83
Beef	0.87	52.13	6.9h	131.17	150.77	2.52	188.97
Beetle/Fly	1.62	13.96	122.87	42.92	26.49	3.08	2.86
CBF	1.19	2.04	518.49	16.43	13.81	8.07	31.56
ChlorineConcentration	1.82	128.56	31.8h	173.86	95.53	1.35	659
Coffee	0.98	4.1	53.3	10.96	11.18	2.67	4.86
Computers	55.6	398.47	34.4h	1049.52	18.88	2.63	117.9
CricketZ	7.84	741.89	40.6h	5.8h	2663.27	28.3	6.96
DiatomSizeReduction	1.02	4.33	749.03	30.04	29.45	6.94	24.93
DistalPhalanxOutlineCorrect	4.05	15.2	1.1h	52.39	12.93	3.45	73.61
Earthquakes	39.75	1152.06	2.7h	2957.36	73.4	2.57	3.27
ECG200	1.12	2.96	325.94	48.34	43.16	16.35	6.74
ECG5000	6.95	36.89	7.5h	600.37	86.38	16.27	44.76
ECGFiveDays	1.31	1.01	24.16	1.38	1.05	1.37	17.51
ElectricDevices	1.7h	782.76	27.4d	5.8h	3.5	26.64	113.73
FaceAll	12.34	155.43	11.3h	1541.8	124.93	9.92	26.54
FaceFour	1.51	17.15	181.63	32.67	21.64	1.91	5.56
FacesUCR	4.83	40.91	10.7h	1265.7	262.05	30.94	30.33
FordA	301.9	4.5h	548.78	10.4h	124.01	2.32	0.01
GunPoint	1.09	1.69	486.48	8.97	8.23	5.31	54.23
Ham	2.07	172.19	2.02h	126.1	60.92	0.73	57.67
HandOutlines	72.88	10.9h	/	1.2h	59.28	0.11	/
Haptics	6.45	2086.17	3.6d	3.2h	1786.05	5.52	27.23
InlineSkate	5.42	2.1h	8.5d	4.2h	2856.09	1.97	48.66
InsectWingbeatSound	7.94	147.67	1.1d	646.49	81.42	4.38	144.77
LargeKitchenAppliances	650.37	1018.59	4.2d	3.9h	21.59	13.72	25.68
Mallat	4.34	419.98	3.9h	2896.15	667.32	6.9	4.87
Meat	1.94	34.62	2975.9	44.02	22.69	1.27	67.6
NonInvasiveFatalECGThorax1	30.34	5.6h	/	11.1h	1317.07	1.99	/
OSULeaf	7.41	381.4	18.02h	1.9h	923.08	17.71	9.61
Phoneme	462.23	2.6h	53.7d	12.7h	98.91	4.82	101.4
RefrigerationDevices	80.87	1.3h	3.5d	2.5h	111.29	1.9	34.08
ShapeletSim	3.24	26.23	28.33	455.23	140.5	17.35	0.06
SonyAIBORobotSurface1	1.15	0.85	271.38	4.19	3.64	4.91	64.77
SonyAIBORobotSurface2	0.93	0.83	292.42	3.78	4.06	4.54	77.36
Strawberry	5.02	93.33	8.4h	235.17	46.84	2.52	128.15
Symbols	1.04	16.17	2315.93	90.43	86.95	5.59	25.61
SyntheticControl	1.91	8.87	1.5h	249.29	130.52	28.11	20.99
ToeSegmentation1	2.04	7.75	1.9h	19.91	9.76	2.57	343.61
TwoLeadECG	1.86	0.91	397.09	20.32	10.92	22.22	19.54
TwoPatterns	7.71	195.12	20.99h	4.96h	2315.95	91.69	4.22
UWaveGestureLibraryY	35.89	1231.53	5.9d	2.2d	5296.18	157.26	2.65
Wafer	6.39	64.15	2.7h	825.96	129.26	12.88	11.58
WormsTwoClass	5.75	2085.67	6.3h	1124.08	195.49	0.54	20.24
Yoga	5.81	333.24	1.2d	2.9h	1796.9	31.79	9.46
Average					488.09	13.84	67.07

Table 3.4: Efficiency of BSPCOVER and ELIS on UCR ARCHIVE

Dataset	ELIS (seconds)				Ours (seconds)				Improvement	
	discover	train	test	total	discover	train	test	total	discover	total
Beef	109.61	24675.86	1.70	24847.17	15.72	114.32	1.13	131.17	6.97	188.42
Beetle/Fly	51.80	70.68	0.39	122.87	14.35	28.12	0.45	42.92	3.61	2.86
Bird/Chicken	60.14	42.46	0.26	102.86	11.14	21.97	0.07	33.18	5.40	3.10
Coffee	50.56	2.73	0.01	53.3	1.96	8.86	0.14	10.96	25.80	4.86
Diatom.	60.42	688.05	0.56	749.03	2.62	26.06	1.36	30.04	23.06	24.93
ECGFiveDays	13.79	9.7	0.67	24.16	0.36	0.48	0.55	1.38	38.31	17.51
FaceFour	117.33	63.62	0.68	181.63	3.06	28.87	0.74	32.67	38.34	5.56
GunPoint	40.99	444.54	0.95	486.48	0.67	8.09	0.21	8.97	61.18	54.23
ItalyPower	0.72	636.7	0.55	637.97	0.06	1.85	0.23	2.14	12.00	298.12
Lighting7	554.88	15838.13	3.68	16396.69	21.4	573.67	1.72	596.79	25.93	27.47
MoteStrain	3.84	1470.2	1.2	1475.24	0.19	3.74	0.74	4.67	20.21	315.90
SonyAIBO1	2.31	268.76	0.31	271.38	0.24	3.34	0.61	4.19	9.63	64.77
Symbols	227.58	2082.63	5.72	2315.93	6.04	91.83	5.17	103.04	37.68	22.48
Trace	374.78	73.11	0.37	448.26	5.92	44.02	0.3	50.24	63.31	8.92
TwoLeadECG	4.45	391.49	1.15	397.09	0.19	19.35	0.78	20.32	23.42	19.54
Average									26.32	70.58

$p = 5$. The model building time is significantly more computationally costly than BSPCOVER's, which is observed in the "discover" and "train" columns of "Ours" of Table 3.4.

3.3.4.3 Cost model

We conducted an experiment to compute the constants $c_{i,i\in 1,2}$ for each factor in a cost model and validate whether it can fit the runtimes of BSPCOVER. As discussed in Section 3.2.4, the efficiency of BSPCOVER is mainly contributed to by three components. The total time cost is estimated as follows. For the illustration purpose, we present only the dataset **Beef** to derive the constants c_i and the same procedure can be utilized to explore c_i for other datasets. The main idea is to compute the constant through modifying one parameter and fixing others for each factor as control variables. For instance, below is the cost model for **Beef**;

$$T_{\text{BSPCOVER}} = c_1 I + c_2 p + \epsilon = 90 \cdot I + 4,667 \cdot p + 27,389 \tag{3.8}$$

where both I and $p \in \mathbb{N}$ and I is often a multiple of 1000.

The standard deviation of c_1, c_2 and ϵ are $2.57, 8.94$, and 14.58; and the root-

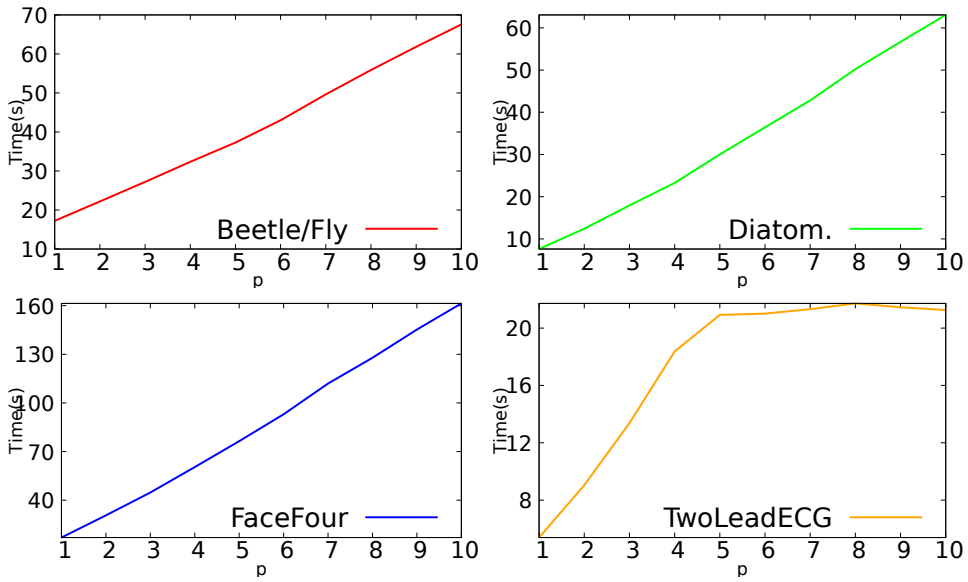


Figure 3.7: Efficiency vs p (of p-Cover) on four UCR datasets

mean-square error of T_{BSPCOVER} is 979.26, less than 1 second. Hence, users can employ the above method to reasonably estimate the runtimes and determine the values of I and p for the datasets.

3.3.5 Experiments on Accuracy

The experiment accuracy results for baselines (with the except of **Random**) are all taken from the paper [2] with a single train/test. The accuracy results for 45 datasets are presented in Table 3.5.

3.3.5.1 Comparison with other methods

While the current most accurate classifier remains COTE, our method is very competitive. In particular, it is close to ST and ResNet, and superior to the others on average. BSPCOVER exhibits the highest accuracy in 11 datasets. Its accuracy is ranked the $3rd$, with a rank mean of 3.78 and the rank of most datasets is top-three among all nine methods. The 1-to-1 wins numbers for RotF, DTW_Rn_1NN, LS, FS, SD, ELIS and Random are all more than half of all the datasets. It is not surprising that BSPCOVER is more efficient than COTE (and all other shapelet-based methods except FS and SD). In other words, BSPCOVER shows that it is possible to make a

small sacrifice in accuracy to achieve a significant improvement in efficiency.

We conducted the Friedman test [69] among all the methods and Wilcoxon signed-rank test between other ten methods and our BSPCOVER. The Friedman test is a well-known non-parametric statistical test, to detect the differences in 45 datasets across 11 methods. Our result is that $p = 0.00$, which is smaller than 0.05. Thus, we can reject the null hypothesis, and there is a difference among these 11 methods. However, we cannot figure out the difference from exactly which methods only according to the Friedman test. We further conducted the Wilcoxon test against all baselines and found out that all results are statistically significant at $p < 0.05$, except **ST**, **ResNet** (larger than 0.05) and **COTE** (based on opposite ranks) from the last row in Table 3.5.

3.3.5.2 Comparison with BSPCOVER-Random

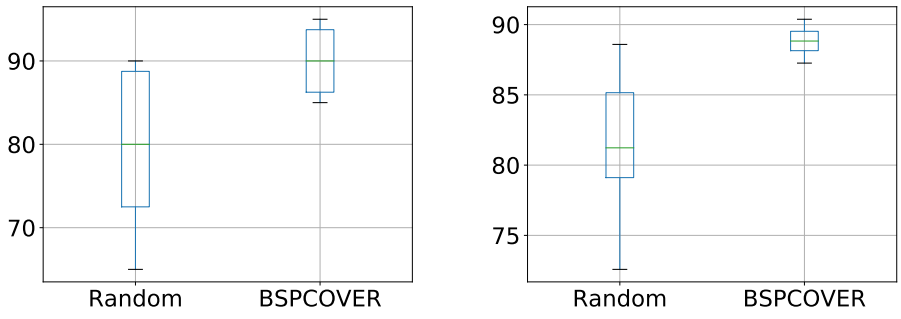
From the last two columns of Table 3.5, we can observe that the results of BSPCOVER are better than those of BSPCOVER-Random in all the datasets, which proves the superiority of our heuristic algorithm. We compared the accuracy of two methods with box plots, generated from ten runs for each dataset. Some representative datasets are shown in Figure 3.8. The box plots show the minimum, first quartile, median, third quartile, and maximum values of four datasets, namely **Beetle/Fly**, **Diatom.**, **FaceFour**, and **TwoLeadECG**. The metrics of the box plot in our BSPCOVER method are better than those of the BSPCOVER-Random method.

3.3.5.3 Accuracy by varying p

To investigate the performance of p-Cover value, a set of experiments is designed to verify the trend between p-Cover value and accuracy. Four datasets, **Beetle/Fly**, **Diatom.**, **FaceFour**, and **TwoLeadECG** are selected. We can observe in Figure 3.9 that as the p-Cover value increases, the accuracy also rises at

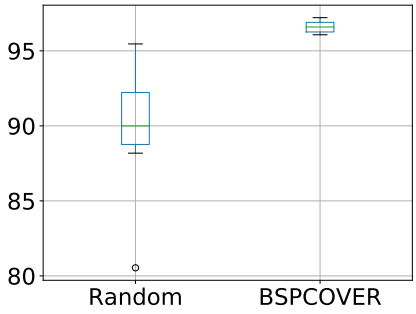
Table 3.5: Accuracy of BSPCOVER and 10 related methods on UCR ARCHIVE

Dataset	RotF	DTW_Rn_1NN	ST	LS	FS	SD	COTE	ELIS	ResNet	Random	BSPCOVER
ArrowHead	73.71	80	73.71	**84.57**	59.43	65.7	81.14	81.43	84.5	75.33	80.57
Beef	86.67	66.67	**90**	86.67	56.67	50.7	86.67	63.33	75.3	54.21	73.33
BeetleFly	**90**	65	**90**	80	70	75	80	85	85	80	**90**
CBF	92.89	99.44	97.44	99.11	94	97.5	99.56	90.44	99.5	92.58	**99.67**
ChlorineConcentration	**84.74**	65	69.97	59.24	54.64	55.3	72.71	27.39	84.4	45.63	61.22
Coffee	**100**	**100**	96.43	**100**	92.86	96.1	**100**	96.43	**100**	93.47	**100**
Computers	70	62.4	73.6	58.4	50	58.8	74	50	**81.5**	57.74	67.2
CricketZ	65.64	73.59	78.72	74.1	46.41	67.3	**81.54**	78.95	81.2	68.91	74.1
DiatomSizeReduction	87.25	93.46	92.48	**98.04**	86.6	89.6	92.81	89.86	30.1	81.69	87.25
DistalPhalanxOutlineCorrect	75.72	72.46	77.54	77.9	75	71.7	76.09	57.83	77.1	77.52	**83.17**
Earthquakes	74.82	72.66	74.1	74.1	70.5	63.6	74.82	77.64	71.2	75.49	**81.68**
ECG200	85	88	83	88	81	81.8	88	80	87.4	85	**92**
ECG5000	94.58	92.51	94.38	93.22	92.27	92.4	**94.6**	72.69	93.4	92.64	94.44
ECGFiveDays	90.82	79.67	98.37	**100**	99.77	95.3	99.88	95.45	97.5	89.95	**100**
ElectricDevices	**78.58**	63.08	74.7	58.75	57.9	59.3	71.33	8.65	72.9	18.19	24.24
FaceAll	91.12	80.77	77.87	74.85	62.6	71.4	**91.78**	75.56	83.9	72.44	76.33
FaceFour	81.82	89.77	85.23	**96.59**	90.91	82	89.77	95.46	95.5	93.32	**96.59**
FacesUCR	80.29	90.78	90.59	93.9	70.59	84.7	94.24	63.63	**95.5**	72.14	78.29
FordA	84.47	66.52	**97.12**	95.68	78.71	77.6	95.68	67.6	92.0	90.12	96.31
GunPoint	92	91.33	**100**	**100**	94.67	93.1	**100**	97.57	99.1	87.38	**100**
Ham	71.43	60	68.57	66.67	64.76	61.9	64.76	63.81	75.7	71.87	**76.19**
HandOutlines	91.08	87.84	**93.24**	48.11	81.08	79.9	91.89	/	91.1	75.28	86.7
Haptics	43.83	41.56	**52.27**	46.75	39.29	35.6	**52.27**	41.56	51.9	41.31	45.13
InlineSkate	37.09	38.73	37.27	43.82	18.91	38.5	**49.45**	35.46	37.3	32.67	38.73
InsectWingbeatSound	63.64	57.37	62.68	60.61	48.94	44.1	**65.25**	59.55	50.7	50.82	57.42
LargeKitchenAppliances	60.8	79.47	85.87	70.13	56	57.1	84.53	33.33	**90**	82.37	86.13
Mallat	94.93	91.43	96.42	95.01	**97.61**	92.6	95.39	81.58	97.2	72.51	76.8
Meat	96.67	93.33	85	73.33	83.33	93.3	91.67	55	**96.8**	70.14	75
NonInvasiveFatalECGThorax1	90.53	82.9	**94.96**	25.9	71.04	81.4	93.13	/	94.5	82.36	91.47
OSULeaf	57.02	59.92	96.69	77.69	67.77	56.6	96.69	76.45	**97.9**	75.39	83.88
Phoneme	12.97	22.68	32.07	21.84	17.35	15.8	**34.92**	15.19	33.4	17.27	20.73
RefrigerationDevices	56.53	44	**58.13**	51.47	33.33	46.1	54.67	40	52.5	52.91	54.67
ShapeletSim	41.11	69.44	95.56	95	**100**	67.2	96.11	**100**	77.9	79.25	84.44
SonyAIBORobotSurface1	80.87	69.55	84.36	81.03	68.55	85	84.53	87.85	**95.8**	80.06	88.35
SonyAIBORobotSurface2	80.8	85.94	93.39	87.51	79.01	78	95.17	93.17	**97.8**	79.93	93.49
Strawberry	97.3	94.59	96.22	91.08	90.27	88.4	95.14	83.85	**98.1**	89.57	94.29
Symbols	79.3	93.77	88.24	93.17	93.37	90.1	**96.38**	78.29	90.6	86.23	93.37
SyntheticControl	97.33	98.33	98.33	99.67	91	98.3	**100**	99.33	99.8	95.57	99.67
ToeSegmentation1	53.07	75	96.49	93.42	95.61	88.2	97.37	**98.24**	96.3	81.71	96.49
TwoLeadECG	97.01	86.83	99.74	99.65	92.45	86.7	99.3	99.82	**100**	92.19	99.65
TwoPatterns	92.8	99.85	95.5	99.33	90.83	98.1	**100**	99.75	**100**	92.64	99.8
UWaveGestureLibraryY	71.44	70.18	73.03	70.3	59.58	67.1	**75.85**	69.32	67.0	57.15	64.01
Wafer	99.45	99.59	**100**	99.61	99.68	99.3	99.98	99.43	99.9	96.49	99.81
WormsTwoClass	68.83	58.44	**83.12**	72.73	72.73	64.1	80.52	71.82	74.7	71.24	74.59
Yoga	82.43	84.3	81.77	83.43	69.5	62.5	87.67	83.9	87.0	81.27	**88.2**
Total best acc	4	1	10	6	2	0	13	2	11	0	11
Ours 1-to-1 Wins	28	29	18	25	40	36	13	34	19	45	-
Ours 1-to-1 Draws	3	2	3	7	1	0	3	1	1	0	-
Ours 1-to-1 Losses	14	14	24	13	4	9	29	8	25	0	-
Rank Mean	6.34	6.68	3.78	5.16	8.43	8.58	3.02	7.46	3.65	8.21	3.78
Wilcoxon Test p-vlaue	0.003	0.001	0.079	0.048	0.00	0.00	0.002	0.00	0.122	0.00	-

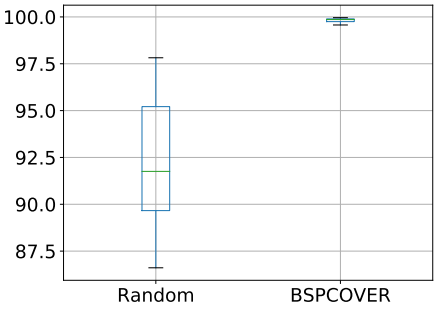


(a) Beetle/Fly

(b) Diatom.



(c) FaceFour



(d) TwoLeadECG

Figure 3.8: Accuracy of Random vs BSPCOVER in box plots on four UCR datasets

the beginning. The accuracy typically stabilizes after p-Cover reaches the optimal value.

3.3.5.4 Accuracy by varying I

The quality of the shapelets discovered can be enhanced by increasing the number of iterations I in model building. Hence, we conducted another experiment that varies I and studied the accuracies of BSPCOVER and the most recent approach, ELIS. We chose four datasets due to their difference in iteration numbers for convergence is not significant. We observe in Figure 3.10 that the accuracy in all four subfigures increases with the iteration number. Our BSPCOVER discovers high-quality shapelets in three datasets, **GunPoint**, **SonyAIBO1**, and **Symbols** and the accuracy stabilizes early.

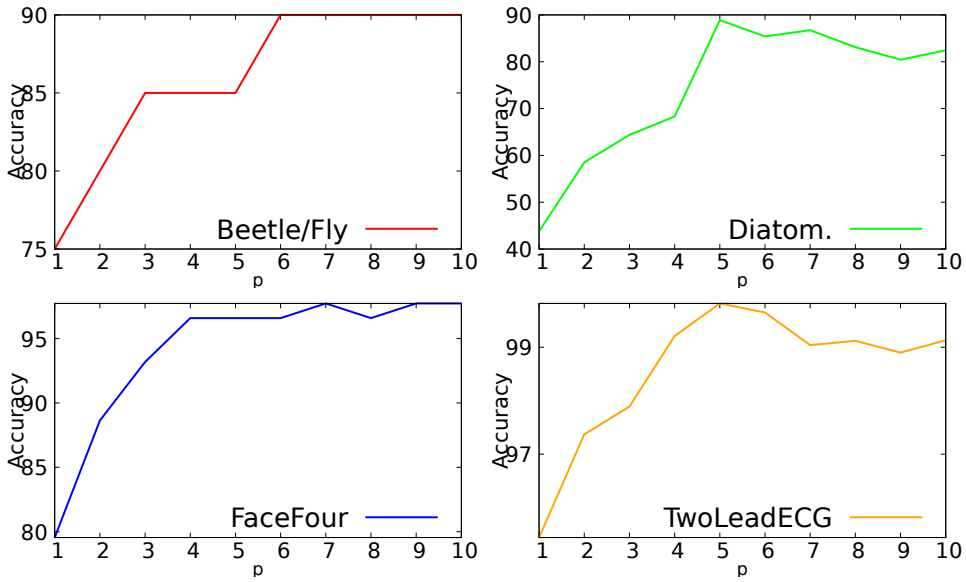


Figure 3.9: Accuracy vs p (of p-Cover) on four UCR datasets

3.3.6 Experiments on Interpretability

One of the strengths of shapelets is their interpretability, which can help practitioners to understand their data. This experiment investigates whether the shapelets discovered by our method are interpretable. We present results from two datasets because they do not require much domain background. We calculate the best matching score (best matching location) of each shapelet in the raw time series and users can exploit them to greedily select a small set of shapelets.

3.3.6.1 Interpretability of ECGFiveDays Shapelets

Figure 3.11 shows how the shapelets discriminate the different classes of the **ECGFiveDays** dataset. ECG stands for Electrocardiography. The time series in Figure 3.11 (a) (b) belong to Class 1 (abnormal ECG) and those in Figure 3.11 (c) (d) belong to Class 2 (normal ECG). Our selected shapelets are colored in green and black for Class 2. More importantly, the shapelets highlight the major different fragments in two classes. (Shapelet 1 is QRS complex and Shapelet 2 is T wave of ECG.) Intuitively, the peak of T wave is larger than that of QRS complex in Class 1. It is indeed called hyperacute T wave in medicine, due to ischemia or hyperkalemia

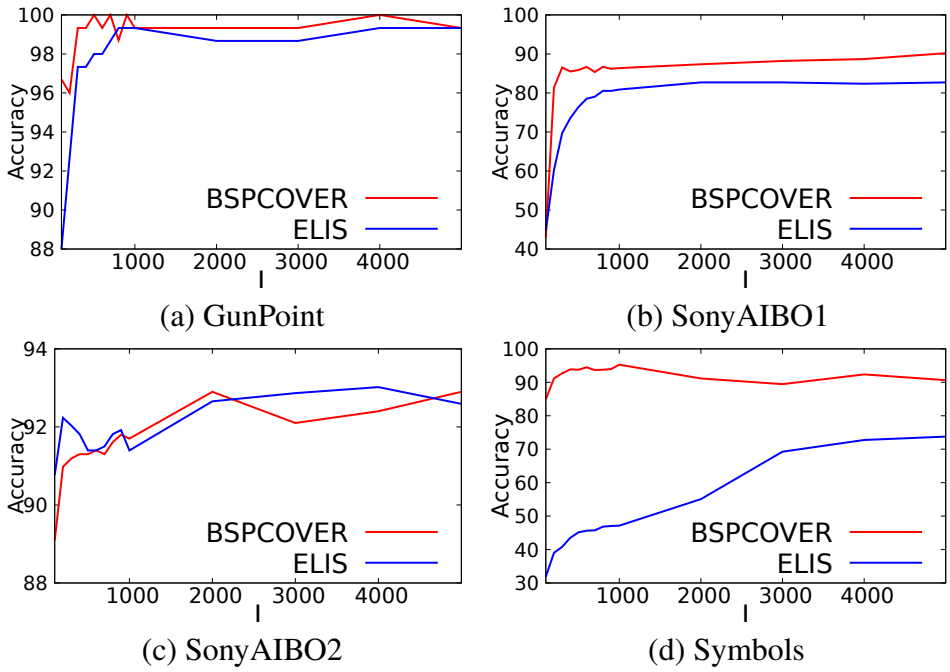


Figure 3.10: Accuracy vs iterations of BSPCOVER and ELIS on four UCR datasets

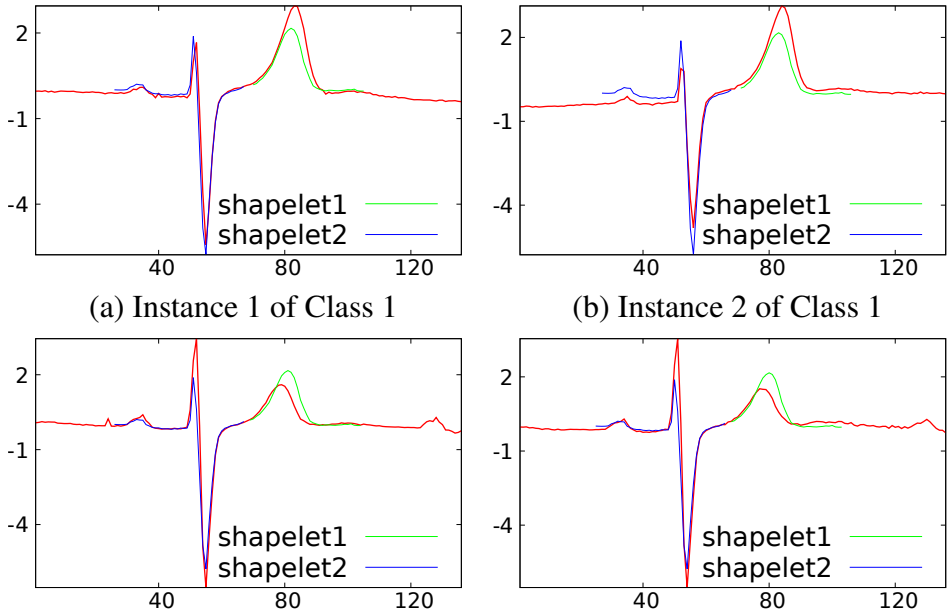
or other diseases.

3.3.6.2 Interpretability of ItalyPowerDemand Shapelets

Another interpretability result comes from the **ItalyPowerDemand** dataset, shown in Figure 3.12. The time series instances in Figure 3.12 (a) (b) belong to Class 1 (summer months, from April to September), and those in Figure 3.12 (c) (d) exist in Class 2 (winter months, from October to March). The most representative shapelet selected by BSPCOVER for Class 1 illustrates the difference between the two classes. The difference is that the power demand in Class 1 is lower than that in Class 2 from 4am to 12pm, which turns out to be the heating in the morning during the winter time. (Air conditioning was still fairly rare in Italy when the data was collected.)

3.4 Chapter Summary

This chapter has proposed an efficient method of shapelet discovery, called BSPCOVER. BSPCOVER has four steps: it transforms raw time series by adopting PAA and SAX techniques; it adopts bloom filters to prune duplicate SAX words and proposes an

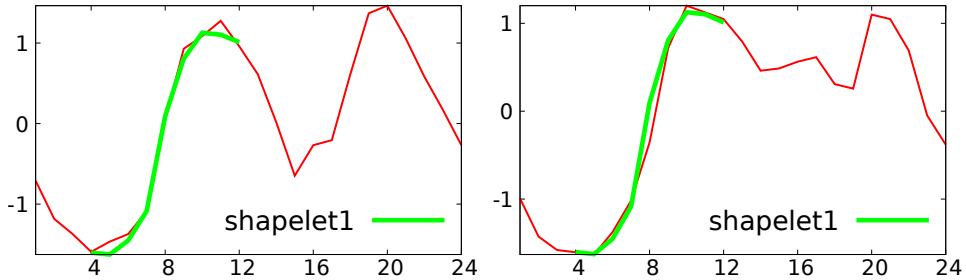


(a) Instance 1 of Class 1 (b) Instance 2 of Class 1

(c) Instance 1 of Class 2 (d) Instance 2 of Class 2

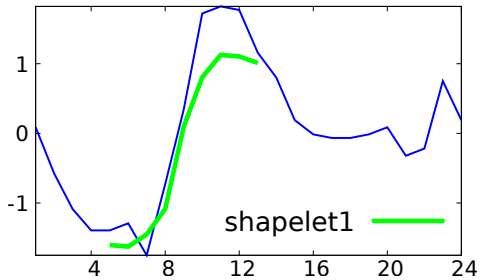
Figure 3.11: **ECGFiveDays** shapelets encodes hyperacute T wave

algorithm to prune similar non-discriminative SAX words; and it computes a subset of shapelet candidates with maximal weights to cover each class p times. Hence, potentially costly learning algorithms are applied to a discriminative subset only. Our experiments verify that BSPCOVER is more efficient than almost all shapelet-based methods (with the sole exception of FS) and its accuracy is comparable to or slightly lower than the most accurate classifier, COTE.

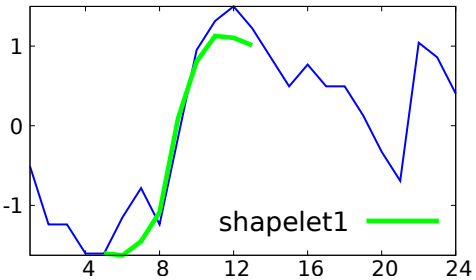


(a) Instance 1 of Class 1

(b) Instance 2 of Class 1



(c) Instance 1 of Class 2



(d) Instance 2 of Class 2

Figure 3.12: A shapelet of **ItalyPowerDemand** highlighting the morning heating demand difference of summer and winter months

Chapter 4

Efficient Shapelet Disocvery Univariate Time Series Classification - A Matrix Profile Approach

In this chapter, we investigate the problem of time series clustering. Time series classification preliminaries and three issues of MP baseline method are introduced in Section 4.1. Section 4.2 proposes the details of our MPS, which is an matrix profile-based shapelet approach for time series classification, for discovering a set of high-quality shapelets. Section 4.3 presents our experimental results on the efficiency and accuracy. Finally, we summarize this chapter in Section 4.4.

4.1 Preliminaries

In this section, we introduce some background knowledge and analyze three issues with the MP baseline method for time series classification.

4.1.1 Terminologies and Notations

In this section, we present some specific terminologies and summarize the notations and their meanings in Table 4.1.

Definition 4.1. Matrix profile. Given a time series T with length N and a specific

Table 4.1: Summary of frequently used notations in MPS

Notation	Meaning
T_C	concatenating all time series instances from class C into one longer time series instance
P	the matrix profile (value)
diff	the difference between matrix profile
dist	the ED distance between time series (subsequences)
label	the label of time series

subsequence length L, the *matrix profile* is a data structure to annotate T,

$$MP(T, L) = (mp_1, mp_2, \cdots, mp_i, \cdots, mp_{N-L+1}) \tag{4.1}$$

where mp_i, $i \in [1, N - L + 1]$ is the nearest neighbor distance of the subsequence $T_{i,i+L-1}$ among the other subsequences with length L in T.[1] □

4.1.2 Three Issues of an MP Baseline Method

Baseline. Foremost, we introduce the main idea of the baseline method of using MP to determine shapelets [29]. The baseline selects the time series subsequence at the location of the largest difference in the MP as the final shapelet for class A (indicated in Figure 1.5). Specifically, Formula 4.2 shows how is the subsequence selected by the baseline method. It finds a time series subsequence $\mathcal{S}$ as "shapelet" has the largest difference between the distances of $\mathcal{S}$ and the concatenated time series T_B and T_A, respectively.

$$\arg\max(\mathsf{diff}(\mathrm{P}_{AB}, \mathrm{P}_{AA})) = \arg\max_{\substack{\mathcal{S} \\ \mathsf{label}(T_B) \neq \mathsf{label}(T_A)}} ||\mathsf{dist}(\mathcal{S}, T_B) - \mathsf{dist}(\mathcal{S}, T_A)||, \tag{4.2}$$

where $\mathcal{S}$ is from class A of length L, starting from $i \in [0, N - L + 1]$.

The idea above can be easily extended to discover the top-k shapelets by using the top-k largest differences. We then illustrate three issues of the baseline method with an example of the dataset ArrowHead in Figure 1.2.

1st issue: Discords as "shapelets". The first issue is that the largest difference indi-

[1] Since the subsequences located near $T_{i,i+L-1}$ may have small distance from $T_{i,i+L-1}$, those subsequences are excluded from finding mp_i.

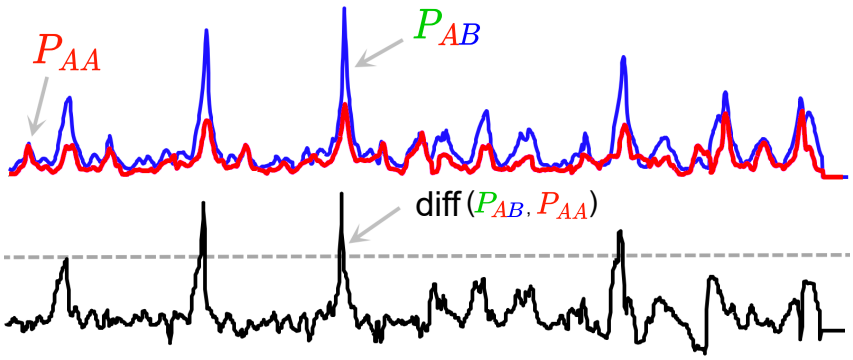


Figure 4.1: An example of matrix profiles and the difference to show the first issue Discords as "shapelets"

cator only considers the difference of distances between $\mathsf{dist}(\mathcal{S}, T_B)$ and $\mathsf{dist}(\mathcal{S}, T_A)$, *i.e.*, $||\mathsf{dist}(\mathcal{S}, T_B) - \mathsf{dist}(\mathcal{S}, T_A)||$ is large. There are two possible scenarios.

- One scenario is the larger $\mathsf{dist}(\mathcal{S}, T_B)$ (discord in T_B) and the smaller $\mathsf{dist}(\mathcal{S}, T_A)$ (motif in T_A). $\mathcal{S}$ appears in T_A but not in T_B, which shows $\mathcal{S}$ can be a shapelet of high quality.
- Another scenario is that $\mathcal{S}$ is a discord in both T_A and T_B, where $\mathsf{dist}(\mathcal{S}, T_A)$ and $\mathsf{dist}(\mathcal{S}, T_B)$ are both large. If the distances can be large, their difference can still be large. This does not necessarily indicate $\mathcal{S}$ can distinguish class A from B but there is simply no subsequence in both classes that is similar to $\mathcal{S}$.

Example 4.1. We observe the largest difference in the below part of Figure 4.1. The value in the P_{AB} (in the above part of Figure 4.1) is the largest one, the value in the P_{AA} (in the above part of Figure 4.1) is however the largest one (the second scenario). That value in P_{AA} shows that the subsequence is a *discord* in the T_A, which contradicts that the subsequence should present widely in class A [14]. □

Assume another two scenarios are with the same difference between P_{AA} and P_{AB} value, and they are both the largest difference, which causes the shapelet determination confusion.

Example 4.2. One scenario is both the discord in class B and class A, the other is a

discord in class B and a motif in class A. We formalize the scenario in the following.

$$\underbrace{||\mathsf{dist}(\mathcal{S}_1, T_B) - \mathsf{dist}(\mathcal{S}_1, T_A)||}_{\mathcal{S}_1 \text{ is a discord both in } T_B, T_A} = \underbrace{||\mathsf{dist}(\mathcal{S}_2, T_B) - \mathsf{dist}(\mathcal{S}_2, T_A)||}_{\mathcal{S}_2 \text{ is a discord in } T_B, \text{ but motif in } T_A}$$

Although the shapelet candidate in the second case (namely, $\mathcal{S}_2$) is better than $\mathcal{S}_1$ under the intuition of shapelet, it is difficult for MP baseline method to determine the final top-1 shapelet through Formula 4.2. □

2nd issue: Classification ineffectiveness. The second issue of the MP baseline method is that the discovered shapelets may only separate few time series instances. The baseline method concatenates all the time series instances into one long time series instance (*e.g.*, T_A, T_B), which shrinks the number of classified instances for the final shapelets. From Formula 4.2, we can further investigate that T_A/T_B (Figure 1.2) is one long concatenated time series instance, which means the discovered shapelets only be aimed at the long time series instance T_A/T_B. However, the shapelets should classify all the time series instances from dataset D. For example, the top-1 shapelet can only classify some of the time series instances, even though the shapelet is the motif in class A and the discord in class B. We can realize that the accuracy of top-k setting to 1 on MP baseline method is much lower than those of 1NN-ED [17] and 1NN-DTW [17] in Table 4.2.

3rd issue: Similar subsequences as shapelets. The final issue is that the MP baseline method *concatenates* the time series of a class into one long time series. The MP is built on the concatenated time series and the baseline method does not necessarily find shapelets that represent the whole class. When such shapelets are used for classification, they could not yield high accuracy.

As the proof in [52], the essence of matrix profile is "*the 1-NN based nonparametric density estimation of time series subsequences*". The proof detail of 1-NN based density estimation, a low-quality estimator, is in the [70].

The pool of shapelet candidates is thus much smaller after concatenation, only one time series. Although the kNN (selecting the top-k shapelets) may alleviate this

Table 4.2: The accuracy of the different top-k Shapelets on MP baseline, 1NN-ED, and 1NN-DTW on four datasets from UCR Archive

The k in top-k	1	2	5	10	20	50	100	ED	DTW
ArrowHead	61.71	64	61.14	65.14	61.28	**65.71**	61.71	**80**	70.29
MoteStrain	69.88	77.47	77.08	**78.59**	77.02	77.39	78.19	**87.79**	83.47
ShapeletSim	52.23	55.56	54.44	58.33	**60.56**	57.77	56.11	53.89	**65**
ToeSegmentation1	66.66	67.1	70.18	68.86	71.49	**72.36**	71.93	67.98	**77.19**

issue, the diversity of shapelets cannot be enhanced since there can be high similar shapelet candidates. It can be observed from Figure 1.2, Figure 1.4, and Figure 1.5. The top-5 shapelet indicators refer to similar subsequences in T_A, which limits its classification effectiveness.

To investigate this possibility of three issues, we study that the accuracies of the baseline's top-k shapelets by tuning k from 1 to 100. The results are reported in Table 4.2. It can be observed that the baseline method of various ks has lower accuracies when compared to simple methods such as 1NN-ED [17] and 1NN-DTW [17]. On the contrary, the shapelets cannot distinguish on average 33% of time series from class A and class B even when k = 100. In all, there are three issues that affect the quality of the discovered shapelets.

4.2 MPS for Shapelets

In this section, we propose a novel matrix profile-based shapelet approach, called MPS. Figure 1.6 shows the overview of MPS.

From ① the multiple samples which are randomly selected from all the time series instances, we employ the ② motifs calculated by the matrix profiles as the initial shapelet candidates from the samples rather than only the difference among matrix profiles of each class. The motifs are the frequent time series subsequences, which meet the part of shapelets' definition, widely existing, but may not appear in one class. We then discover the motifs existing in one class through pruning with motifs/discords in other classes as the final shapelets. The pool of shapelet

candidates is large. ③ A bloom filter is simultaneously constructed for the SAX representations of the initial shapelet candidates ④ to prune the similar candidates from the inter (other) classes. We propose ⑤ three utility functions (from intra class motifs, inter classes motifs/discords, and intra class time series instances) to score the shapelet candidates for computing the final top-k shapelets. The distances among motif candidates (intra class), motifs/discords (inter classes), and time series instances (intra class) are embedded into the utility functions based on the definition of shapelets.

We develop the essence of shapelet discovery in the following.

$$\arg\max_{\mathcal{S}} \sum_{\substack{T',T\in D \\ \mathsf{label}(T')\neq\mathsf{label}(T)}} ||\mathsf{dist}(\mathcal{S},T') - \mathsf{dist}(\mathcal{S},T)|| \tag{4.3}$$

We consider the original time series instances, rather than one concatenated long time series for each class.

4.2.1 Shapelet Candidates Pool Generation

We present the details of shapelet candidates pool generation as follows. We exploit the matrix profile (MP) [29] to obtain the shapelet candidates (motifs and discords) in a bagging way [33]. In particular, we randomly select some, not all time series instances for each class to form a sample and concatenate them into one long time series. Then, we calculate the MP of each long time series to obtain sufficient motifs and discords. A bloom filter can efficiently prune a large number of candidates with high similarities.

Algorithm 4.1 details the shapelet candidates pool generation procedure. We randomly select some time series instances without replacement from the raw training set in one class C to generate a sample Q_C (Lines 2-5). We concatenate the selected time series instances into one long time series T_{Q_C} in the sample (Line 6). Matrix profile is employed to calculate the motif and discord under the specific length of candidates (Lines 7-8). There are many other samples to be taken from the same

previous processes (Line 4). We then obtain the initial motif pool and discord pool for class C as the candidates (Line 11). A bloom filter $\mathcal{BF}$ (Line 10) for class C is constructed under the SAX representation [34] for the further pruning process.

Next, we define the C-motif to efficiently prune the similar candidates which exist in the inter classes, namely $\mathcal{C} \setminus \{C\}$.

Definition 4.2. **C-motif**. Given a time series dataset D with the label set $\mathcal{C}$ and the generated motifs of class C, the C-motif is the remaining motifs after pruning the similar motifs existing in other classes $\bar{\mathcal{C}} = \mathcal{C} \setminus \{C\}$. Specifically, a bloom filter $\mathcal{BF}$ is utilized in the following,

$$\bigvee_{\bar{\mathcal{C}}=\mathcal{C}\setminus\{C\}} \Big(\mathcal{BF}_{\bar{\mathcal{C}}}.\mathsf{lookup}\big(\mathsf{SAX}(e)\big) == \mathsf{True}\Big) \tag{4.4}$$

where $\mathsf{SAX}()$ is the function to generate the motif subsequence e into SAX representation. □

Definition 4.3. **C-discord**. Given a time series dataset D with the label set $\mathcal{C}$ and the generated discords of class C, the C-discord is the remaining discords after pruning the similar discords existing in other classes $\bar{\mathcal{C}} = \mathcal{C} \setminus \{C\}$. Specifically, a bloom filter $\mathcal{BF}$ is utilized in the following,

$$\bigvee_{\bar{\mathcal{C}}=\mathcal{C}\setminus\{C\}} \Big(\mathcal{BF}_{\bar{\mathcal{C}}}.\mathsf{lookup}\big(\mathsf{SAX}(e)\big) == \mathsf{True}\Big) \tag{4.5}$$

where $\mathsf{SAX}()$ is the function to generate the discord subsequence e into SAX representation. □

We further elaborate the details of the C-motif pruning process in Algorithm 4.1 (Lines 14-17), we calculate all the SAX representations for the bloom filters of inter other classes $\bar{\mathcal{C}} = \mathcal{C} \setminus \{C\}$ (Line 15). One possible scenario is that the query result is "True", which means that the candidate exists in other classes $\bar{\mathcal{C}}$, we then remove the candidate e (Line 16) and update the motif set Ω^{motif} (Line 17). Otherwise, the candidate is maintained. Similarly, we define the C-discord and do the pruning.

Algorithm 4.1: Generation of the pool for shapelet candidates

Input: Dataset D with $\mathcal{C}$, sample number Q_N, sample size Q_S, candidate lengths ϕ

Output: Candidates pool Ω

1 {Generating initial candidates pool}
2 **foreach** $C \in \mathcal{C}$ **do**
3 Initialize $\Omega_C = \varnothing$;
4 **foreach** $q \in Q_N$ **do**
5 $Q_C = D.\mathsf{Random}(Q_S)$;
6 $T_{Q_C} = Q_C.\mathsf{concatenate}()$;
7 **foreach** $l \in \phi$ **do**
8 $(\Omega_C^{motif}, \Omega_C^{discord}) = \mathrm{MP}(T_{Q_C}, l)$;
 /* Bloom filters to prune similar motifs/discords */
9 $\mathcal{BF}_C.\mathsf{add}(\mathsf{SAX}(\Omega_C^{motif}, \Omega_C^{discord}))$;
10 $\Omega_C += (\Omega_C^{motif}, \Omega_C^{discord})$;
11 {Pruning similar candidates in pool}
12 **foreach** $C \in \mathcal{C}$ **do**
13 **foreach** $e \in \Omega_C^{motif}$ **do**
14 **if** $\bigvee_{\bar{\mathcal{C}}=\mathcal{C}\setminus\{C\}} \Big(\mathcal{BF}_{\bar{C}}.\mathsf{lookup}\big(\mathsf{SAX}(e)\big) == \mathsf{True}\Big)$ **then**
15 $\Omega_C^{motif}.\mathsf{subtract}(e)$;
16 $\Omega_C.\mathsf{update}(\Omega_C^{motif})$;
17 **foreach** $e \in \Omega_C^{discord}$ **do**
18 **if** $\bigvee_{\bar{\mathcal{C}}=\mathcal{C}\setminus\{C\}} \Big(\mathcal{BF}_{\bar{C}}.\mathsf{lookup}\big(\mathsf{SAX}(e)\big) == \mathsf{True}\Big)$ **then**
19 $\Omega_C^{discord}.\mathsf{subtract}(e)$;
20 $\Omega_C.\mathsf{update}(\Omega_C^{discord})$;
21 **return** Ω

Complexity analysis. The time complexity of Lines 2-11 and Lines 13-21 are $O(|\mathcal{C}| \cdot |Q_N| \cdot |\phi| \cdot N^2)$ and $O(|\mathcal{C}| \cdot |Q_N|)$, respectively. Hence, the time complexity of Algorithm 4.1 is $O(N^2)$, where N is the length of time series.

4.2.2 Top-k Shapelet Generation

From the candidate pool, we score candidates and then select top-k according to their quality, which are similar to the candidates of its class (small intra class distance) and dissimilar to the candidates of other classes (large inter class distance). In addition to

the candidates' perspective, we also consider the time series instances from the intra class to avoid the scenario in Example 4.2. The shapelet determination confusion is thus settled with the final score for each candidate.

We propose three sigmoid utility functions for scoring the candidates from the intra class and inter classes, respectively. We explain the details of selecting shapelets for class C. The motif candidates in class C are scored by Formula 4.6, which means the total distance between one of the motif candidate and other candidates. When the total distance of the candidate is small, it indicates that the motif candidate can represent other motif candidates in class C (one shapelet's attribute is *widely existing in intra class*). That is, the smaller the total distance among the motif candidate, the likelier the candidate a true shapelet. We then employ the motifs/discords from the inter classes $\{\mathcal{C} \setminus C\}$ to score motif candidates (from class C) by Formula 4.7. The large value computed by Formula 4.7 implies that the motif candidate (from class C) has a large distance with motifs/discords (from $\{\mathcal{C} \setminus C\}$), which signifies another shapelet's attribute, *far away from the other classes*. For making shapelet determination distinct after two previous scoring processes, we incorporate the distance between motif candidates and the raw time series instances also from class C into the score. Formula 4.8 is designed for the characteristics of shapelets that *the time series instances in the same class should be closer to the shapelet*.

Intra scoring. We first calculate the utility (Formula 4.6) between each motif candidate and other motif candidates from the motif pool in the same class.

Definition 4.4. Intra-class Utility. The intra-class utility is the distance between the *motif* candidate Can_i and other *motif* candidates $\sum_{j=1}^{|\Omega_C|} Can_j$ is denoted as,

$$\mathcal{U}_{intra}(Can_i) = \frac{1}{1 + e^{-\sum_{j=1}^{|\Omega_C|} ||Can_i - Can_j||_2^2}} \tag{4.6}$$

where both Can_i and Can_j are from intra class C. □

Inter scoring. The following is the inter scoring. The difference is that Can_j is the motifs/discords from other classes.

Definition 4.5. Inter-class Utility. The inter-class utility is the distance between the *motif* candidate Can_i and other *motif/discord* candidates Can_j is denoted as,

$$\mathcal{U}_{inter}(Can_i) = \frac{1}{1 + e^{-\sum_{j=1}^{|\Omega_{\{\mathcal{C}\setminus C\}}|} ||Can_i - Can_j||_2^2}} \tag{4.7}$$

where Can_i from class C and Can_j from inter classes $\{\mathcal{C} \setminus C\}$. □

Intra instances scoring. We further calculate the distance between the shapelet candidates and all the time series instances from the same class to reduce the shapelet determination confusion by Formula 4.2.

Definition 4.6. Intra Instances Utility. The utility is the distance between the *motif* candidate Can_i and time series T_j is denoted as,

$$\mathcal{U}_{D_C}(Can_i) = \frac{1}{1 + e^{-\sum_{j=1}^{|D_C|} ||Can_i - T_j||_2^2}} \tag{4.8}$$

where both Can_i and T_j are from class C. □

Algorithm 4.2 is the pseudo-code of scoring top-k shapelets. First, we initialize the set of shapelets $\mathcal{S}$ (Line 1). For each class C, we first calculate the utility $Score_C$ of motif candidates for each length based on Formula 4.6 (Lines 2-6). We then subtract $Score_C$ from the score in the inter classes with their motifs (Line 8) and discords (Line 9). The subtraction can maintain the attribute of *far away from the inter classes*. The score (Line 11) between the motif candidates and time series instances from class C is added for the sorting under the ascending order (Line 12). The addition also preserves another attribute of *instances closer to the shapelet*. According to the utility of each motif candidate in C, we select the candidates with top-k minimum score as for the final shapelets (Line 13).

Algorithm 4.2: Top-k shapelet generation

Input: Dataset D with $\mathcal{C}$, candidates pool Ω, candidate lengths ϕ, final shapelet number k for each class

Output: Final shapelets $\mathcal{S}$

Initialize $\mathcal{S} = \varnothing$;

for $C \in \mathcal{C}$ **do**

  Initialize $Score_C = \varnothing$;

  for $l \in \phi$ **do**

    /* Score candidates in intra class */

    $Score_C += \bigcup_{i=1}^{|\Omega_C^{motif}|} \mathcal{U}_{intra}(Can_i)$; // Formula 4.6

    /* Score candidates in inter classes */

    $Score_C -= \bigcup_{i=1}^{|\Omega_C^{motif}|} \mathcal{U}_{inter}(Can_i)$; // Formula 4.7

    $Score_C -= \bigcup_{i=1}^{|\Omega_C^{motif}|} \mathcal{U}_{inter}(Can_i)$; // Formula 4.7

    /* Score candidates with instances */

    $Score_C += \bigcup_{i=1}^{|\Omega_C^{motif}|} \mathcal{U}_{D_C}(Can_i)$; // Formula 4.8

  $\Omega_C^{motif}.\mathsf{sort}(Score_C)$;

  $\mathcal{S}_C^k = \Omega_C^{motif}.\mathsf{pop}(k)$;

return $\mathcal{S} = \bigcup_{C=1}^{|\mathcal{C}|} \mathcal{S}_C^k$

Complexity analysis. The time complexity of Algorithm 4.2 is $O(|Q_N|^2)$. The full complexity is $O\Big(|\mathcal{C}| \cdot \big[|\phi| \cdot \big(|Q_N|^2 + |\mathcal{C}| \cdot |Q_N|^2 + |Q_N| \cdot |D_C|\big) + |Q_N| \cdot \log(|Q_N|) + k\big]\Big)$.

Remarks. The final shapelets are then employed to transform the original time series [16], [21], [43] for learning a classification model. In this thesis, we adopt SVM with a linear kernel for the classification.

4.3 Experimental Results

In this section, we report the experiments of MPS with 10 related methods on the widely-used UCR datasets. In particular, we report the accuracy and efficiency of MPS on the 46 datasets and illustrate the interpretability of the discovered shapelets

with one case study.

4.3.1 Experimental Settings

We have implemented the proposed method in PYTHON. All the experiments were conducted on a machine with a Xeon E5-2630 v4 @ 2.2GHz (2S/10C) / 256GB RAM / 128GB SWAP, running on CentOS 7.6 (64-bit). We can easily set the parameters for the bloom filters of each dataset because they are space efficient.

The protocol used in the previous work [2], [27] have been followed. The 10 compared methods are namely Rotation Forest [2] and 1NN-DTW [68], and many other shapelet-based methods including ST [16], LTS [9], Fast shapelets [10], SD [38] COTE [21] ELIS [1], the current state-of-the-art efficient method BSPCOVER [27] and the MP baseline method BASE [29]. We omit the details of each method in this thesis. Interested readers may refer to the original papers for details.

Some parameters adopted in the experiments are listed in the following. For the fairness, we set the shapelet number as 5 in both BASE and MPS. The various lengths of shapelet candidates ranging from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ are demonstrated. That number means a ratio of subsequence's length to the original time series' length (*e.g.*, 0.2 means that the subsequence's length is 20% of the original time series' length). The sample number Q_N and sample size Q_S are selected from $\{10, 20, 50, 100\}$ and $\{2, 3, 4, 5, 10\}$, respectively.

4.3.2 Experiments on Efficiency

From the previous work, we observe that COTE and ST are slower than LTS [2], and the efficiency of ELIS is about two orders of magnitude faster than that of LTS [1], and BSPCOVER is 70x faster than ELIS [27]. Thus, we compare BASE and BSPCOVER with MPS to investigate the efficiency.

Comparison with BASE and BSPCOVER . The total running time of BASE, BSPCOVER, and our MPS, and the improvement of BASE over MPS and that of MPS over

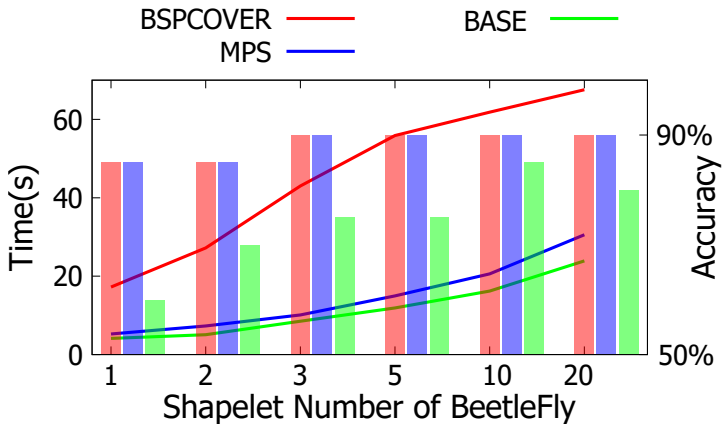


Figure 4.2: Efficiency vs Top-k (shapelet number) of BSPCOVER, BASE, MPS methods on BeetleFly

BSPCOVER, reported in Table 4.3 (The unit of the numbers is second).

MPS is consistently faster than BSPCOVER on these 46 datasets, on average, about 21x faster in terms of total running time. The accuracy of MPS is higher in 23 datasets, but lower in 16 datasets. As for BASE, it is only slightly faster than MPS (1.4x on average), however, its accuracy is lower in 41 datasets Table 4.4. Hence, MPS discovers high-quality shapelets to enhance the final accuracy with an efficiency similar to BASE.

Efficiency by varying the shapelet number k. Figure 4.2 and Figure 4.3 exhibit the total running time (line chart) and the accuracy (bar chart) with the increase of shapelet number k on two datasets (BeetleFly and TwoLeadECG), and three methods, namely BASE, MPS, and BSPCOVER.

The accuracy of BASE on both two datasets is lower than that of MPS. MPS's accuracy is similar to BSPCOVER 's. The number of shapelets contributes to the accuracy at the beginning and then stabilizes. For the efficiency on BeetleFly, MPS and BASE maintain an approximate linear relationship with k. The runtime of BSPCOVER is clearly larger than the other two methods, whereas the differences between the runtimes of BASE and MPS are not significant. Regarding TwoLeadECG, a similar trend can be observed when $k \leq 10$ among three methods. Nevertheless,

Table 4.3: Efficiency of MPS and related methods on UCR Archive (second as unit) and the speedup

Dataset	BASE	BSPCOVER	MPS	Speedup BASE vs MPS	Speedup MPS vs BSPCOVER
ArrowHead	7.65	55.57	12.79	1.67	4.35
Beef	10.56	131.17	18.15	1.72	7.23
BeetleFly	16.19	42.92	20.58	1.27	2.09
CBF	4.53	16.43	6.03	1.33	2.73
ChlorineConcentration	29.17	173.86	35.48	1.22	4.90
Coffee	5.15	10.96	7.98	1.55	1.37
Computers	103.63	1049.52	148.57	1.43	7.06
CricketZ	641.89	20993.38	886.47	1.38	23.68
DiatomSizeReduction	11.87	30.04	16.20	1.36	1.85
DistalPhalanxOutlineCorrect	12.67	52.39	19.08	1.51	2.75
Earthquakes	178.06	2957.36	202.45	1.14	14.61
ECG200	9.17	48.34	16.78	1.83	2.88
ECG5000	30.01	600.37	45.01	1.50	13.34
ECGFiveDays	1.06	1.38	1.25	1.18	1.10
ElectricDevices	202.86	20851.50	289.04	1.42	72.14
FaceAll	108.63	1541.80	156.23	1.44	9.87
FaceFour	9.59	32.67	11.13	1.16	2.94
FacesUCR	5.19	1265.70	8.36	1.61	151.45
FordA	236.45	37481.21	298.71	1.26	125.48
GunPoint	2.28	8.97	3.50	1.54	2.56
Ham	11.49	126.10	25.78	2.24	4.89
HandOutlines	607.26	4340.86	712.89	1.17	6.09
Haptics	504.48	11523.26	673.91	1.34	17.10
InlineSkate	993.56	15060.30	1123.69	1.13	13.40
InsectWingbeatSound	169.25	646.49	208.55	1.23	3.10
ItalyPowerDemand	0.72	0.91	3.85	1.26	4.23
LargeKitchenAppliances	412.52	13974.80	538.50	1.31	25.95
Mallat	135.05	2896.15	195.66	1.45	14.80
Meat	8.85	44.02	11.59	1.31	3.80
NonInvasiveFatalECGThorax1	15385.63	40125.40	19886.26	1.29	2.02
OSULeaf	99.48	6753.46	128.89	1.30	52.40
Phoneme	3586.33	45767.83	4325.86	1.21	10.58
RefrigerationDevices	1258.34	8871.13	1563.59	1.24	5.67
ShapeletSim	30.08	455.23	44.57	1.48	10.21
SonyAIBORobotSurface1	2.39	4.19	3.09	1.29	1.36
SonyAIBORobotSurface2	1.65	3.78	2.59	1.57	1.46
Strawberry	15.64	235.17	21.88	1.40	10.75
Symbols	10.11	90.43	18.63	1.84	4.86
SyntheticControl	5.36	249.29	7.01	1.31	35.58
ToeSegmentation1	1.62	19.91	3.26	2.01	6.11
TwoLeadECG	8.27	20.32	9.32	1.13	2.18
TwoPatterns	149.16	17891.24	188.95	1.27	94.69
UWaveGestureLibraryY	956.34	193667.34	1238.47	1.30	156.38
Wafer	50.49	825.96	68.48	1.36	12.06
WormsTwoClass	305.49	1124.08	389.56	1.28	2.89
Yoga	207.58	10593.18	274.07	1.32	38.65
Average				**1.40**	**21.64**

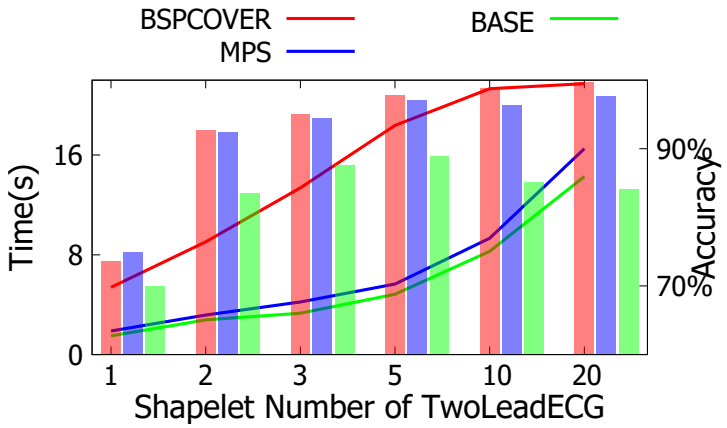


Figure 4.3: Efficiency vs Top-k (shapelet number) of BSPCOVER, BASE, MPS methods on TwoLeadECG

the total time stabilizes when $k > 10$ in BSPCOVER and still grows rapidly in BASE, MPS. Li et al. [27] consider that all the shapelet candidates have been determined when $k = 10$ in BSPCOVER.

4.3.3 Experiments on Accuracy

The experiment accuracy results for 9 methods (with the exception of BASE) are all taken from the papers [2], [27]. The results of MPS and BASE are the mean values of 5 runs and the standard deviations of all the datasets are less than 0.01. We provide some highlights of accuracy results with some selected datasets of different types, *e.g.*, Image, Sensor, Simulated, and Motion.

Comparison with other methods. The overall accuracy results for 46 datasets are presented in Table 4.4. We observe that the overall accuracy of MPS is ranked the $2nd$ among all 11 methods from Table 4.4, where the $1st$ is the ensemble method COTE. COTE contains a collection of classifiers and MPS can be certainly incorporated into COTE. Furthermore, MPS performs the best on 10 datasets (still ranked $2nd$), which is only lower than COTE, but much higher than DTW_Rn_1NN, LTS, FS, SD, ELIS, and BASE. MPS achieves clearly much higher accuracy in some datasets, such as RefrigerationDevices and SonyAIBORobotSurface1. The 1-to-1-Losses

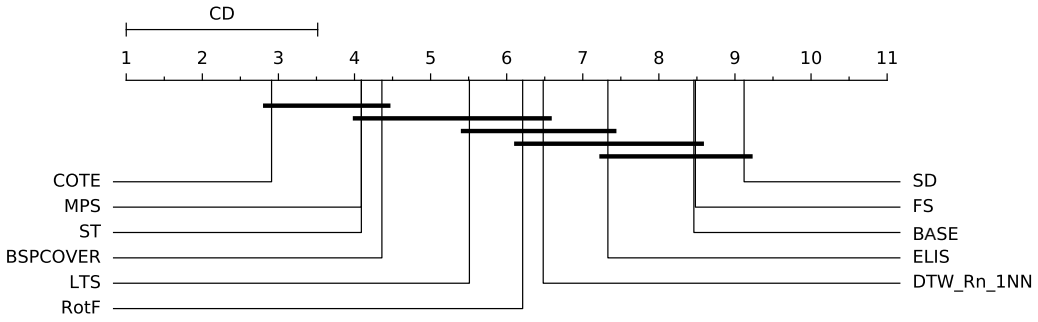


Figure 4.4: Critical difference diagram of the pairwise statistical comparison of 11 methods on the UCR Archive

datasets on accuracy are marginally lower than ST (*e.g.*, CricketZ and FAceAll) and BSPCOVER (*e.g.*, Haptics and InlineSkate).

Friedman test and Wilcoxon test. The Friedman test [69], a non-parametric statistical test, and Wilcoxon-signed rank test with Holm's α (5%) [71] is taken for all the methods.

The Friedman test is to detect the differences among 46 datasets across 11 methods. The statistical significance p-value is 0.00, smaller than $\alpha = 0.05$. Thus, we reject the null hypothesis and a significant difference among these 11 methods.

We then conduct the post-hoc analysis among all the methods. The results are visualized by a critical difference diagram[2] in Figure 4.4. We observe that MPS significantly outperforms other approaches except for COTE, ST, and BSPCOVER.

Accuracy by varying the shapelet numbers k. We investigate the performance on different number of shapelet from $\{1, 2, 5, 10, 20\}$. The impact of different shapelet numbers on the final accuracy of MPS on four datasets, Arrowhead, MoteStrain, ShapeltSim, and ToeSegmentation1.

Figure 4.5 shows the accuracy by varying the shapelet numbers. Four different datasets show the different trends, which guide the appropriate shapelet number of the dataset. For example, the accuracy stabilizes in ToeSegmentation1 through all the

[2]A thick horizontal line groups a set of classifiers that are not significantly different.

Table 4.4: Accuracy of MPS and related methods on UCR Archive

Dataset	RotF	DTW_Rn_1NN	ST	LTS	FS	SD	COTE	ELIS	BSPCOVER	BASE	MPS
ArrowHead	73.71	80	73.71	84.57	59.43	65.7	81.14	81.43	80.57	61.14	85.14
Beef	86.67	66.67	90	86.67	56.67	50.7	86.67	63.33	73.33	50	73.33
BeetleFly	90	65	90	80	70	75	80	85	90	75	90
CBF	92.89	99.44	97.44	99.11	94	97.5	99.56	90.44	99.67	68	99.78
ChlorineConcentration	84.74	65	69.97	59.24	54.64	55.3	72.71	27.39	61.22	54.66	63.41
Coffee	100	100	96.43	100	92.86	96.1	100	96.43	100	95.14	100
Computers	70	62.4	73.6	58.4	50	58.8	74	50	67.2	66.8	70.5
CricketZ	65.64	73.59	78.72	74.1	46.41	67.3	81.54	78.95	74.1	37.44	78.46
DiatomSizeReduction	87.25	93.46	92.48	98.04	86.6	89.6	92.81	89.86	87.25	89.22	88.89
DistalPhalanxOutlineCorrect	75.72	72.46	77.54	77.9	75	71.7	76.09	57.83	83.17	78.83	83.67
Earthquakes	74.82	72.66	74.1	74.1	70.5	63.6	74.82	77.64	81.68	81.99	81.99
ECG200	85	88	83	88	81	81.8	88	80	92	88	88
ECG5000	94.58	92.51	94.38	93.22	92.27	92.4	94.6	72.69	94.44	92.34	94.44
ECGFiveDays	90.82	79.67	98.37	100	99.77	95.3	99.88	95.45	100	77.82	99.88
ElectricDevices	78.58	63.08	74.7	58.75	57.9	59.3	71.33	8.65	24.24	53.99	55.47
FaceAll	91.12	80.77	77.87	74.85	62.6	71.4	91.78	75.56	76.33	70.18	76.36
FaceFour	81.82	89.77	85.23	96.59	90.91	82	89.77	95.46	96.59	81.82	92.78
FacesUCR	80.29	90.78	90.59	93.9	70.59	84.7	94.24	63.63	78.29	67.61	80.58
FordA	84.47	66.52	97.12	95.68	78.71	77.6	95.68	67.6	96.31	63.32	84.78
GunPoint	92	91.33	100	100	94.67	93.1	100	97.57	100	82.67	100
Ham	71.43	60	68.57	66.67	64.76	61.9	64.76	63.81	76.19	68.57	72.38
HandOutlines	91.08	87.84	93.24	48.11	81.08	79.9	91.89	/	86.7	73.8	89.9
Haptics	43.83	41.56	52.27	46.75	39.29	35.6	52.27	41.56	45.13	30.19	43.51
InlineSkate	37.09	38.73	37.27	43.82	18.91	38.5	49.45	35.46	38.73	21.27	43.82
InsectWingbeatSound	63.64	57.37	62.68	60.61	48.94	44.1	65.25	59.55	57.42	17.63	56.52
ItalyPowerDemand	97.28	95.53	94.75	96.02	91.74	92	96.11	96.57	96.5	92.63	96.6
LargeKitchenAppliances	60.8	79.47	85.87	70.13	56	57.1	84.53	33.33	86.13	57.6	85.34
Mallat	94.93	91.43	96.42	95.01	97.61	92.6	95.39	81.58	76.8	90.54	94.69
Meat	96.67	93.33	85	73.33	83.33	93.3	91.67	55	75	98.33	93.33
NonInvasiveFatalECGThorax1	90.53	82.9	94.96	25.9	71.04	81.4	93.13	/	91.47	56.74	92.06
OSULeaf	57.02	59.92	96.69	77.69	67.77	56.6	96.69	76.45	83.88	57.44	71.49
Phoneme	12.97	22.68	32.07	21.84	17.35	15.8	34.92	15.19	20.73	18.41	28.43
RefrigerationDevices	56.53	44	58.13	51.47	33.33	46.1	54.67	40	54.67	49.87	78.33
ShapeletSim	41.11	69.44	95.56	95	100	67.2	96.11	100	84.44	54.44	84.33
SonyAIBORobotSurface1	80.87	69.55	84.36	81.03	68.55	85	84.53	87.85	88.35	87.35	98.5
SonyAIBORobotSurface2	80.8	85.94	93.39	87.51	79.01	78	95.17	93.17	93.49	82.78	91.71
Strawberry	97.3	94.59	96.22	91.08	90.27	88.4	95.14	83.85	94.29	87.6	96.72
Symbols	79.3	93.77	88.24	93.17	93.37	90.1	96.38	78.29	93.37	69.45	94.1
SyntheticControl	97.33	98.33	98.33	99.67	91	98.3	100	99.33	99.67	94.67	99.67
ToeSegmentation1	53.07	75	96.49	93.42	95.61	88.2	97.37	98.24	96.49	70.18	96.49
TwoLeadECG	97.01	86.83	99.74	99.65	92.45	86.7	99.3	99.82	99.65	88.85	97.1
TwoPatterns	92.8	99.85	95.5	99.33	90.83	98.1	100	99.75	99.8	91.5	99.05
UWaveGestureLibraryY	71.44	70.18	73.03	70.3	59.58	67.1	75.85	69.32	64.01	53.81	65.21
Wafer	99.45	99.59	100	99.61	99.68	99.3	99.98	99.43	99.81	96.24	99.51
WormsTwoClass	68.83	58.44	83.12	72.73	72.73	64.1	80.52	71.82	74.59	42.54	73.48
Yoga	82.43	84.3	81.77	83.43	69.5	62.5	87.67	83.9	88.2	70.53	85.73
Total best acc	8	1	10	5	2	0	13	3	9	2	10
MPS 1-to-1 Wins	30	34	20	26	42	42	13	35	23	41	-
MPS 1-to-1 Draws	2	3	3	5	0	0	4	0	7	2	-
MPS 1-to-1 Losses	14	9	23	15	4	4	29	11	16	3	-

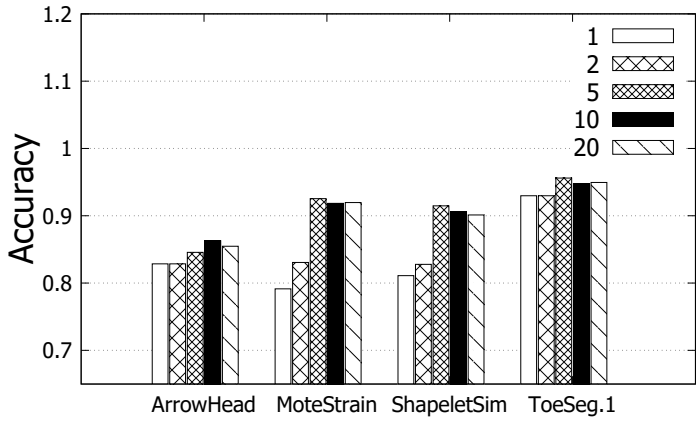


Figure 4.5: Accuracy by varying five kinds of shapelet's numbers on four UCR datasets

shapelet numbers and peaks at 5. Thus, the shapelet number of ToeSegmentation1 is 5. The accuracy rises rapidly with the increase of the shapelet number from 1 to 5 on MoteStrain and then remains stable. Therefore, its shapelet number is set to 5.

4.3.4 Experiments on Interpretability

We further investigate the shapelets' interpretability, a strength of shapelets. We compare the shapelet discovered by our MPS and BSPCOVER in the following on ItalyPowerDemand.

Case study: ItalyPowerDemand. We use ItalyPowerDemand[3] as an example, to show the effectiveness of the discovered shapelet. As the name indicates, ItalyPowerDemand describes the power demand classified by summer and winter in Italy.

From two left subfigures of Figure 4.6, the shapelet $\mathcal{S}$ (green color) demonstrates the difference between two classes, which is similar to the shapelet learned by BSPCOVER (two right subfigures in Figure 4.6). Specifically, both shapelets can catch the power demand in winter (Class 2) is higher corresponding to the heating. The differences between the shapelets discovered by MPS and BSPCOVER are quite minor

[3] http://www.timeseriesclassification.com/description.php?Dataset=ItalyPowerDemand

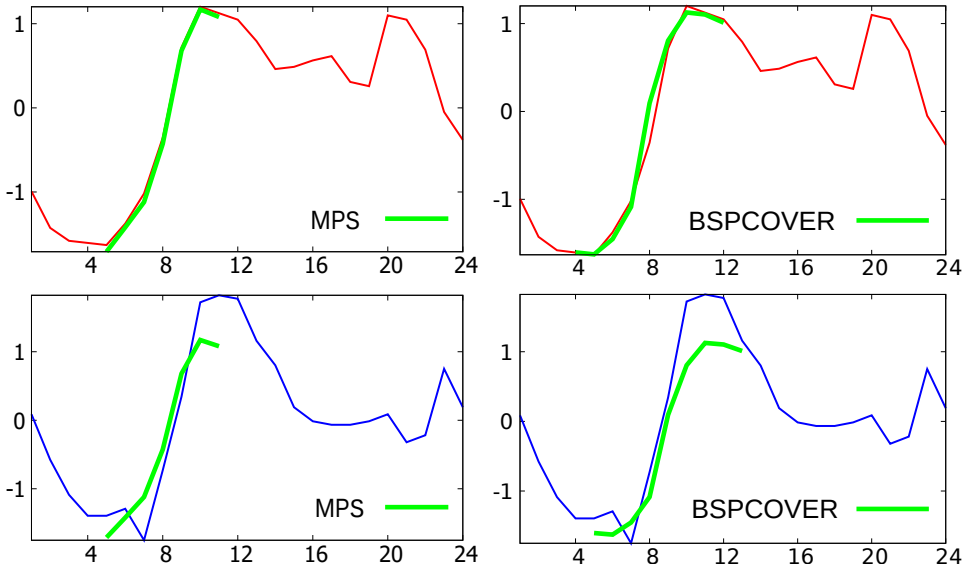


Figure 4.6: Different shapelets (green) from MPS and BSPCOVER of ItalyPowerDemand highlighting the morning heating demand difference of summer (red) and winter (blue) months

but MPS is faster than BSPCOVER 4 times.

4.4 Chapter Summary

This chapter has presented a study on the matrix profile-based shapelet approach for time series classification. We first analyze the inherent three issues of the MP baseline method [29] for time series classification. To address these issues, we proposed MPS, a novel efficient matrix profile-based shapelet approach for time series classification. We obtain the motifs with the matrix profile (MP) for time series instances in a bagging way, which guarantees the quantity and diversity of shapelet candidates pool. We then propose a heuristic algorithm under the utility function to score the final shapelets. The experiment results show that our MPS is superior to the other 10 compared methods on both efficiency and accuracy. We employ the ItalyPowerDemand dataset to illustrate the interpretability of shapelets.

Chapter 5

ShapeNet: A Shapelet-Neural Network Approach for Multivariate Time Series Classification

In this chapter, we investigate the problem of multivariate time series classification. The studied problem and multivariate time series classification problem statement are introduced in Section 5.1. Section 5.2 proposes the details of ShapeNet, which is a shapelet-neural network approach for multivariate time series classification, for discovering a set of high-quality shapelets for model building. Section 5.3 presents our experimental results. Finally, we summarize this chapter in Section 5.4.

5.1 Problem Statement

In this section, we present the problem statement.

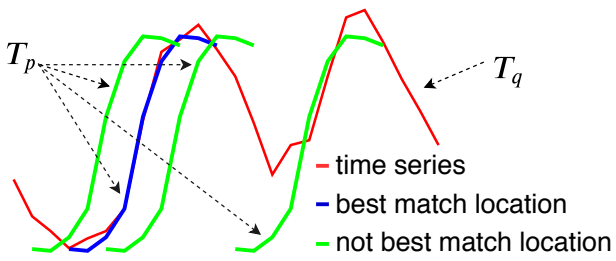


Figure 5.1: An illustration of the best match location of a subsequence in a time series

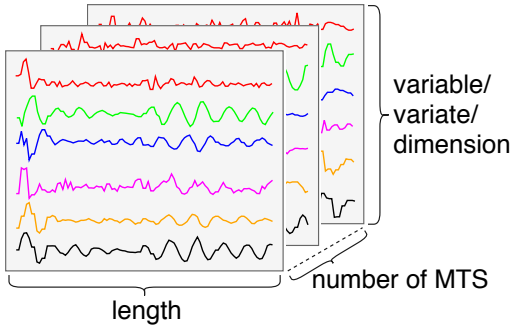


Figure 5.2: An illustration of a multivariate time series dataset — extracted from Basicmotions [42]

Intuitively, dist is the distance of the shorter sequence T_p to the most similar subsequence in T_q, as shown in Figure 5.1.

The apparent difference between univariate and multivariate time series is that there are multiple observations at each timestamp in multivariate time series. An example of the multivariate time series dataset, called Basicmotions from [42], is shown in Figure 5.2. While the definition of shapelet can be naturally extended for multivariate time series, the candidates of shapelet of multivariate time series are voluminous, which makes the classification problem more difficult.

Example 5.1. In Figure 5.2, six variables ($V = 6$) are recorded by an accelerometer and a gyroscope. In the dataset, there are four classes, *i.e.*, $|\mathcal{C}| = 4$, namely standing, walking, running, and playing badminton. Each class is accompanied with 10 training cases (instances) and 10 test cases. Thus, the overall number M of instances is 80. The length (N) of each time series is 100. It is ineffective and inefficient to compute all the distances between each time series and numerous subsequences in the dataset. □

Problem statement. Given a multivariate time series dataset $\mathbb{D}$, consisting of M multivariate time series instances $\mathbb{T}_1, \mathbb{T}_2, \cdots, \mathbb{T}_M$ with V variables, this thesis investigates a shapelet-based classifier. □

5.2 Shapelet Neural Network Approach for MTS classification

In this section, we propose a shapelet-neural network approach, namely ShapeNet. Specifically, we present multi-length-input dialted casual CNN, the cluster-wise triplet loss function, and multivariate shapelet transformation.

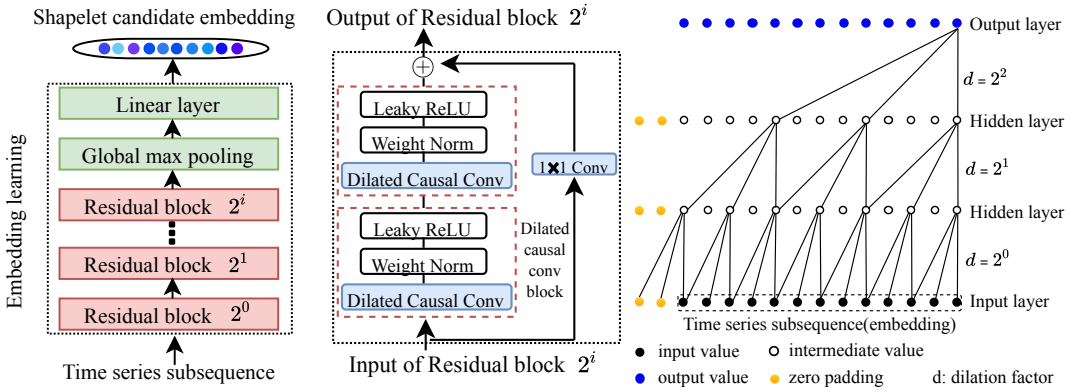
5.2.1 Multi-length-input Dilated Causal CNN (Mdc-CNN)

Shapelet candidates are initially all time series subsequences of different lengths. We use sliding windows (the data shown in the cylinders of Figure 1.7) of discrete sizes to generate the candidates. Our target is to embed all the shapelet candidates from the original space into a new unified space.

Design rationale. ShapeNet adopts a few existing studies as its building blocks. First, the dilated causal convolutional neural network (Dc-CNN) [40] is employed to learn a new representation of time series subsequences. The effectiveness of the dilated causal network has been proved for sequence modeling tasks by [39]. The dilated convolution is utilized to modify the receptive field of the convolution. The causal convolution is designed such that the future data do not impact the learning of the past data.

Second, although the output can be of the same length as the input, Dc-CNN cannot handle inputs of various lengths. Thus, we propose to introduce a global max pooling layer and a linear layer, which are stacked on top of the last Dc-CNN layer, to embed all shapelet candidates into the unified space (indicated by the green boxes in Figure 1.7). We call it *Multi-length-input Dilated Causal CNN (Mdc-CNN)*.

Mdc-CNN architecture. Mdc-CNN is further illustrated in Figure 5.3. Figure 5.3(a) shows that the encoder has $i + 1$ layers of residual blocks, where 2^i is the dilation



(a) Encoder architecture (b) Residual block of Fig. 5.3(a) (c) Dilated causal conv layer of Fig. 5.3(b)

Figure 5.3: An elaboration of the Multi-length-input dilated causal Convolutional Neural Network (Mdc-CNN)

factor, and the global max pooling layer and linear layer are stacked on top of the residual blocks. The input of the encoder is the time series subsequences of various lengths and variables, and the output is their unified representation. We call the output *shapelet candidate embedding*. Figure 5.3(b) presents the residual block with two identical subblocks, and a dilated causal convolution block. Figure 5.3(c) presents a dilated causal convolution example with dilation factor $d = 2^0, 2^1, 2^2$. Further details of Figure 5.3(b) and Figure 5.3(c) can be found in [39].

Following the standard practice (*e.g.*, [72][41]), Mdc-CNNs use shared weights for training models of shapelet candidates of different lengths and variables, as noted on the left side of Figure 1.7. The settings of Mdc-CNN are presented in Section 5.3.2.

5.2.2 Unsupervised Representation Learning

We next explain how the Mdc-CNN networks are trained in an unsupervised manner. There have been several loss functions for unsupervised learning, such as word2vec [73], image similarity [74], and face recognition [41]. In [74] and [41], only one positive sample and one negative sample are considered, whereas, in [60] and [73], one positive and several negative samples are considered. We recall that

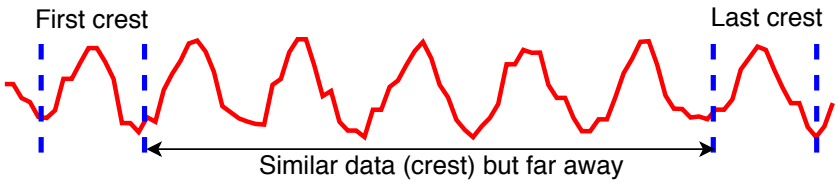


Figure 5.4: An example from Basicmotions of the violation of the second requirement of word2vec: subsequences that are far away but have a small distance between them

Franceschi et al. [60] followed the principle from word2vec [73], which makes the assumption that the representation of a word should meet two requirements: (i) the representation should be close to those near its context [75], and (ii) it should be distant from those in a randomly chosen context, since they are probably different from the original word's context.

The objectives of learning/training (similar to word2vec) are to ensure that similar time series obtain similar representations and vice versa. However, ① the second requirement of the word2vec's assumption does not always hold in the context of time series. For example, one variable of the walking class in the Basicmotions dataset is shown in Figure 5.4. We can easily observe that some crests of the waveform are far away but not distant from each other. ② Only one positive sample is included in a batch to train the network, which is often unstable in the context of the representation learning of shapelets. ③ The distances between negative (positive) samples were not considered before. Figure 5.5 shows the loss in using the original triplet loss [60] to learn shapelet representation. It can be noted that while the loss has slightly declined, it is unstable and hardly converges.

Cluster-wise triplet loss function. In this thesis, we propose a cluster-wise triplet loss function that takes *multiple positive and negative samples* and *the distance among positives (negatives)* as input. For simplicity, we take two clusters to demonstrate our loss function. Specifically, the set of all possible triplets in the training set $\mathcal{T}$ is

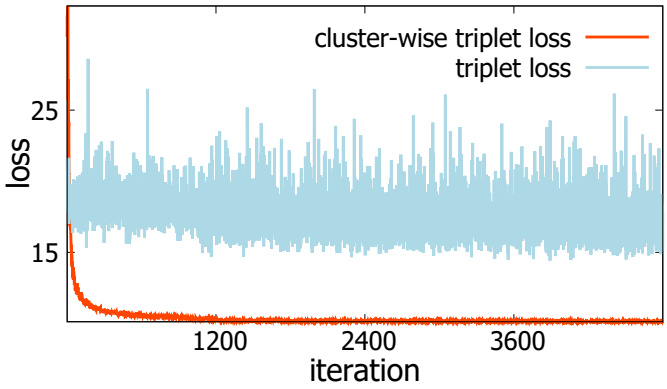


Figure 5.5: A comparison between our cluster-wise triplet loss (multiple positives and multiple negatives, both with intra distances) and original triplet loss (one positive and multiple negatives without intra distance) on ArticularyWordRecognition [42]

defined as follows:

$$\left(x, \boldsymbol{x}^+, \boldsymbol{x}^-\right) \in \mathcal{T},$$

where x is the anchor shapelet candidate, $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ denote the set of positive and negative samples of size K^+ and K^-, respectively.

The number of triplet $(x, \boldsymbol{x}^+, \boldsymbol{x}^-)$ in some real-world datasets is large, and it is computationally prohibitive and sub-optimal to use all the triplets for training. Instead, we conduct triplet sampling. The details of our triplet sampling are presented in Sectuon 5.2.4.

First, we denote the normalized distance of the positive (negative) samples from the anchor as $\mathcal{D}_{AP}$ ($\mathcal{D}_{AN}$), we have the following formula:

$$\mathcal{D}_{AP} + \mu < \mathcal{D}_{AN}, \tag{5.1}$$

where μ is a margin that is enforced between positive and negative samples. Suppose squared Euclidean distance is adopted. $\mathcal{D}_{AP}$ and $\mathcal{D}_{AN}$ can then be defined as follows.

$$\mathcal{D}_{AP} = \frac{1}{K^+} \sum_{i=1}^{K^+} ||f(x) - f(x_i^+)||_2^2 \tag{5.2}$$

and

$$\mathcal{D}_{AN} = \frac{1}{K^-} \sum_{i=1}^{K^-} ||f(x) - f(x_i^-)||_2^2, \tag{5.3}$$

where $f(\cdot) \in \mathbb{R}^z$ is the representation embedded by Mdc-CNN, and z is the length of the embedding.

In addition to the distances between the anchor and the positive (negative) samples, the distances among the positive (negative) samples are included and should be small (large). The maximum distance among all positive (negative) samples is presented in Equation 5.4 (Equation 5.5).

$$\mathcal{D}_{pos} = \max_{i,j \in (1,K^+) \wedge i<j} \{||f(x_i^+) - f(x_j^+)||_2^2\} \tag{5.4}$$

and

$$\mathcal{D}_{neg} = \max_{i,j \in (1,K^-) \wedge i<j} \{||f(x_i^-) - f(x_j^-)||_2^2\} \tag{5.5}$$

The intra-sample loss is defined as follows:

$$\mathcal{D}_{intra} = \mathcal{D}_{pos} + \mathcal{D}_{neg} \tag{5.6}$$

Putting these together, we propose the *cluster-wise triplet loss function for the triplets* for our model in Equation 5.7, to train the network under an unsupervised fashion.

$$\mathcal{L}\left(f(x), f(\boldsymbol{x}^+), f(\boldsymbol{x}^-)\right) = \log \frac{\mathcal{D}_{AP} + \mu}{\mathcal{D}_{AN}} + \lambda \mathcal{D}_{intra} \tag{5.7}$$

where λ is a hyperparameter.

5.2.3 Differentiation of the Loss Function

In order to compute the derivative of Equation 5.7, all the involved functions of the model should be differentiable. Unfortunately, the maximum function of Equation 5.4 and Equation 5.5 in the full paper are not continuous and differentiable. We therefore introduce a differentiable approximation to the maximum function [76].

For the sake of organizational clarity, we use $\mathcal{D}_{i,j}^+$ and $\mathcal{D}_{i,j}^-$ to represent $||f(x_i^+) -$

$f(x_j^+)||_2^2$ and $||f(x_i^-) - f(x_j^-)||_2^2$, respectively.

$$\mathcal{D}_{pos} \approx \tilde{\mathcal{D}}_{pos} = \frac{\sum_{i=1}^{K^+}\sum_{j=1}^{K^+} \mathcal{D}_{i,j}^+ \cdot e^{\alpha \cdot \mathcal{D}_{i,j}^+}}{\sum_{i=1}^{K^+}\sum_{j=1}^{K^+} e^{\alpha \cdot \mathcal{D}_{i,j}^+}} \tag{5.8}$$

and

$$\mathcal{D}_{neg} \approx \tilde{\mathcal{D}}_{neg} = \frac{\sum_{i=1}^{K^-}\sum_{j=1}^{K^-} \mathcal{D}_{i,j}^- \cdot e^{\alpha \cdot \mathcal{D}_{i,j}^-}}{\sum_{i=1}^{K^-}\sum_{j=1}^{K^-} e^{\alpha \cdot \mathcal{D}_{i,j}^-}}, \tag{5.9}$$

where $\alpha > 0$ in Equation 5.8 and Equation 5.9 yields a smooth maximum approximation.

The gradients of overall maximum distance are presented in Equation 5.10 and Equation 5.11.

$$\frac{\partial \tilde{\mathcal{D}}_{pos}}{\partial \mathcal{D}_{i,j}^+} = \frac{e^{\alpha \cdot \mathcal{D}_{i,j}^+}(1 + \alpha(\mathcal{D}_{i,j}^+ - \tilde{\mathcal{D}}_{pos}))}{\sum_{i=1}^{K^+}\sum_{j=1}^{K^+} e^{\alpha \cdot \mathcal{D}_{i,j}^+}} \tag{5.10}$$

$$\frac{\partial \tilde{\mathcal{D}}_{neg}}{\partial \mathcal{D}_{i,j}^-} = \frac{e^{\alpha \cdot \mathcal{D}_{i,j}^-}(1 + \alpha(\mathcal{D}_{i,j}^- - \tilde{\mathcal{D}}_{neg}))}{\sum_{i=1}^{K^-}\sum_{j=1}^{K^-} e^{\alpha \cdot \mathcal{D}_{i,j}^-}} \tag{5.11}$$

Thus, the gradients of Equation 7 in the full paper with respect to $f(x), f(x_i{}^+), f(x_i{}^-)$ are as follows:

$$\frac{\partial \mathcal{L}}{\partial f(x)} = \frac{2\sum_{i=1}^{K^+} ||f(x) - f(x_i^+)||_2}{\sum_{i=1}^{K^+} ||f(x) - f(x_i^+)||_2^2} - \frac{2\sum_{i=1}^{K^-} ||f(x) - f(x_i^-)||_2}{\sum_{i=1}^{K^-} ||f(x) - f(x_i^-)||_2^2} \tag{5.12}$$

$$\frac{\partial \mathcal{L}}{\partial f(x_i{}^+)} = \frac{2\sum_{i=1}^{K^+} ||f(x_i^+) - f(x)||_2}{\sum_{i=1}^{K^+} ||f(x) - f(x_i^+)||_2^2} + \sum_{j=1}^{K^+} \frac{\partial \tilde{\mathcal{D}}_{pos}}{\partial \mathcal{D}_{i,j}^+} \cdot 4||f(x_i^+) - f(x_j^+)||_2 \tag{5.13}$$

$$\frac{\partial \mathcal{L}}{\partial f(x_i{}^-)} = \frac{2\sum_{i=1}^{K^-} ||f(x) - f(x_i^-)||_2}{\sum_{i=1}^{K^-} ||f(x) - f(x_i^-)||_2^2} + \sum_{j=1}^{K^-} \frac{\partial \tilde{\mathcal{D}}_{neg}}{\partial \mathcal{D}_{i,j}^-} \cdot 4||f(x_i^-) - f(x_j^-)||_2 \tag{5.14}$$

Since Equation 7 is differentiable, we use back propagation over the entire neural

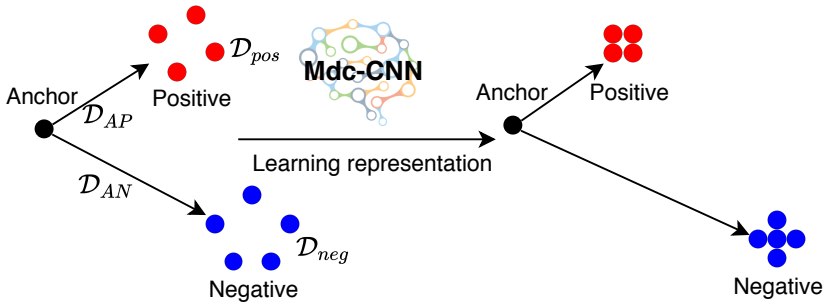


Figure 5.6: An illustration of the effect of training a model using the cluster-wise triplet loss function, positives are closer to each other and the anchor, negatives are closer to each other but farther from the anchor

network based upon minibatch stochastic gradient descent together with Adam optimizer [77] to optimize our ShapeNet parameters.

Discussions. There are two main advantages of our cluster-wise triplet loss function over the previous ones [60]. Firstly, it accelerates convergence and improves stability through not only selecting both multiple positive and negative shapelet candidates but also minimizing the distance among positive/negative shapelet candidates, which have not been considered before. Secondly, the important property of shapelets is that a shapelet is a subsequence (the anchor) of a time series such that most of the time series in one class (positives) are close to it, while most of the time series from other classes (negatives) are far away from it.

Example 5.2. Equation 5.7 is illustrated in Figure 5.6. Two clusters of our cluster-wise triplet loss are illustrated in this example. The triplet loss function both minimizes the distance between the anchor and all positive samples, and the distance among all positive (negative) samples, and maximizes the distance between the anchor (positive) and all negatives. □

5.2.4 Data Preprocessing

In this subsection, we present the details for preparing the multivariate time series data for Mdc-CNN. Specifically, we present the shapelet candidate generation and

Algorithm 5.1: Shapelet candidate generation

Input: MTS dataset $\mathbb{D} = \mathbb{T}^{M \times V \times N}$, sliding window size ϕ
Output: Shapelet candidates Ω

```
for l in φ do
    Initialize Ω_l = ∅ ;
    for m = {1, 2, ··· , M} do
        for v = {1, 2, ··· , V} do
            for i = {1, 2, ··· , N − l + 1 } do
                e = T^v(i, i + l − 1) ;
                Ω_l = Ω_l + e ;
return Ω
```

the triplets selection for the unsupervised representation learning.

Shapelet candidate generation. We apply sliding windows of different sizes to generate abundant shapelet candidates from the original multivariate time series [34][49]. A variate label is then annotated to each candidate, for shapelet transformation in Section 5.2.5. Thus, there are two labels for each shapelet candidate: one for the variable and one for the class of the time series. The shapelet candidate generation procedure is summarized in Algorithm 5.1.

Triplet selection. The numbers of triplets of some real-world datasets are large, and it is computationally prohibitive and sub-optimal to use all the triplets for training. Instead, we conduct triplet sampling.

Specifically, we construct the triplet tuple $(x, \boldsymbol{x}^+, \boldsymbol{x}^-)$, namely $\left(x, \bigcup_{i \in [1,K^+]} (x_i^+), \bigcup_{i \in [1,K^-]} (x_i^-)\right)$ from shapelet candidates at the beginning of each iteration as follows. **I.** We do the clustering on the candidates by kmeans [78] to obtain Y clusters. We randomly select one candidate as the anchor sample x. **II.** Then, from the same cluster, top K^+ other shapelet candidates nearest to the anchor are chosen as positive samples $\boldsymbol{x}^+$. **III.** For the negative samples $\boldsymbol{x}^-$, we randomly pick candidates from other clusters in proportion. The generalization of this procedure to mini-batch training is straightforward, and thus, we omit the details. An example of the triplet selection

Algorithm 5.2: Selection of triplet (APN)

Input: Shapelet candidates Ω
Output: $(x, \boldsymbol{x}^+, \boldsymbol{x}^-)$

1 $\bigcup_{i=1}^{Y} \Omega^i \leftarrow \mathsf{kmeans}(\Omega)$;
2 **for** $i = \{1, 2, \cdots, Y\}$ **do** // for each cluster
3 {Anchor selection}
4 $x \leftarrow \Omega^i.\mathsf{Random}()$;
5 $\Omega^i = \Omega^i \setminus \{x\}$;
6 {Positive selection}
7 **for** $k = \{1, 2, \cdots, K^+\}$ **do**
8 $x^+ = \Omega^i.\mathsf{Top}(x)$;
9 $\Omega^i = \Omega^i \setminus \{x^+\}$;
10 $\boldsymbol{x}^+ = \boldsymbol{x}^+ \cup \{x^+\}$;
11 k++ ;
12 {Negative selection in proportion}
13 **for** $j = \{1, 2, \cdots, Y\} \setminus \{i\}$ **do**
14 **for** $k = \{1, 2, \cdots, \left\lceil \frac{K^-}{Y-1} \right\rceil\}$ **do**
15 $x^- = \Omega^j.\mathsf{Random}()$;
16 $\Omega^i = \Omega^i \setminus \{x^-\}$;
17 $\boldsymbol{x}^- = \boldsymbol{x}^- \cup \{x^-\}$;
18 k++ ;
19 **return** $(x, \boldsymbol{x}^+, \boldsymbol{x}^-)$

process is shown in Figure 5.7. Algorithm 5.2 presents the pseudo-code of triplet selection.

Algorithm 5.2 details the pseudo-code of triplet selection. In Line 1, the shapelet candidates are clustered. This avoids selecting positive (negative, respectively) samples that are highly similar to each other, but gives little information for training. The three parts of Algorithm 5.2 are corresponding to the selection of anchor (Lines 4−5), and positive and negative samples (Lines 7−11 and Lines 13−18), respectively.

Example 5.3. After shapelet candidate generation, we have some candidates with different lengths (the leftmost part of Figure 5.7). For each length, we use a clustering technique to divide them into several groups (4 groups in this example). The following processes are the same for each cluster, and we take cluster 1 as an illustration in Figure 5.7. We arbitrarily choose one time series subsequence (shapelet candidate)

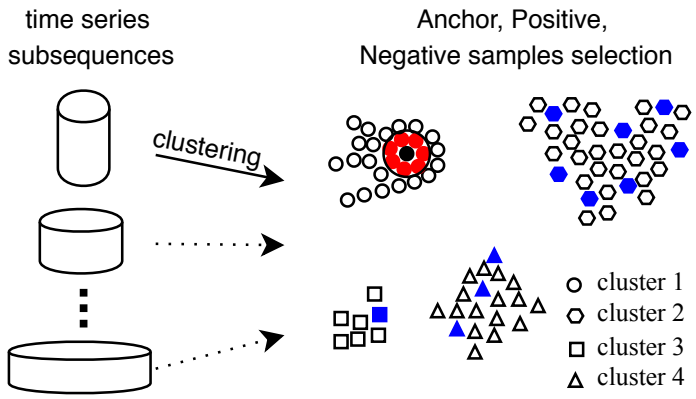


Figure 5.7: An illustration of triplet sampling, black for Anchor, red for Positive samples, blue for Negative samples [best viewed in color]

as the anchor (the solid black circle). We obtain 8 closest candidates as positive samples (the solid red circle). Then, negative samples are selected from other clusters in proportion, specifically, $7, 1, 3$ negative samples (the blue solid hexagon, square, triangle) in these three clusters, respectively. □

After we train the network with the triplets using the cluster-wise triplet loss function, we use the network to embed all the other shapelet candidates.

Analysis. We can find that all elements, namely the anchor, positives, and negatives in the tuple are selected at each batch. As the training goes on, our network minimizes the loss of a batch of triplet tuples at each iteration, which can be approximately regarded as minimizing all the triplets. Thus, although we do not consider the loss of all triplets, the effectiveness of our network can be improved through sampling in practice.

5.2.5 Multivariate Shapelet Transformation

After determining the unified representation of shapelet candidates, we propose to select high-quality and diversified candidates as final shapelets. Finally, we adopt the procedure of shapelet transformation for multivariate time series, then apply a classic classifier to solve the multivariate time series classification problem.

Determining final shapelets. By following previous subsections, all the candidates are embedded into a unified space. It allows us to simply employ a clustering method (*e.g.*, kmeans) to obtain Y clusters of the shapelet candidates. We propose a utility (Equation 5.15) to rank the candidates that are nearest to the cluster centroids. The first component of Equation 5.15 is the size of the candidate's cluster. A large cluster means that it represents many candidates. The second component is the candidate's distance to other candidates in other clusters. A large distance shows that the candidate is different from others:

$$\begin{aligned}\mathcal{U}(f(x_i)) =& \beta \cdot \frac{\log(\mathsf{size}(f(x_i)))}{\log(\max_{i=1}^{Y}(\mathsf{size}(f(x_i))))} \\ &+ (1-\beta)\frac{\log \sum_{j=1}^{Y} ||f(x_i) - f(x_j)||_2^2}{\log(\max_{i=1}^{Y}(\sum_{j=1}^{Y} ||f(x_i) - f(x_j)||_2^2))}\end{aligned} \tag{5.15}$$

where $\beta \in [0, 1]$.

We select the top-k candidates among all Y clusters according to Equation 5.15 and retrieve the original time series subsequences as the *final shapelets*, denoted as $\mathcal{S}_k$.

Multivariate Shapelet Transformation. MST is first mentioned in [37] and the following is our formal definition of it.

Definition 5.1. Multivariate shapelet transformation. Multivariate shapelet transformation is a method to transform a multivariate time series $\mathbb{T}_m$ into a new data space $(d_{m,1}, d_{m,2}, \cdots, d_{m,k})$ by calculating the distances with a set of final shapelets $\mathcal{S}_k$, denoted as $d_{m,j} = \mathsf{dist}(T_m^v, S_j)$, where $S_j \in \mathcal{S}_k$, $T_m^v \in \mathbb{T}_m$, and the variable of S_j and T_m^v is the same. □

Example 5.4. An example of MST is shown in Figure 5.8. The leftmost plot exhibits an instance with six variables from the Basicmotions dataset. Two shapelets, S_1 and

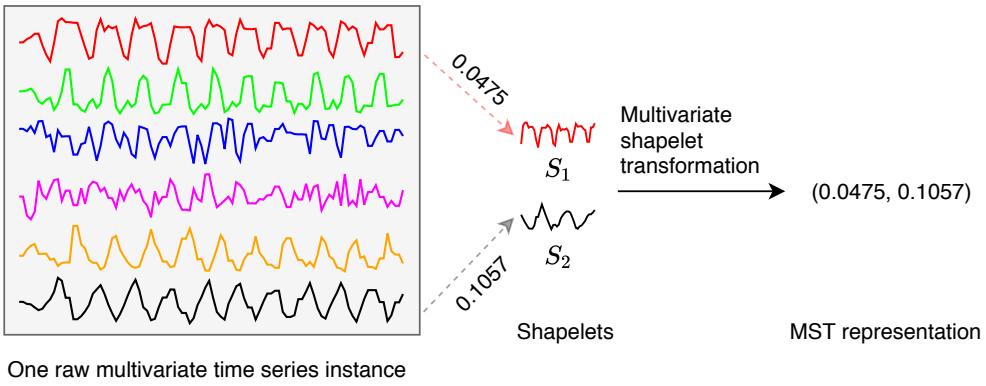


Figure 5.8: An illustration of transforming an MTS instance into the MST representation

S_2, are in the middle. For MST, we calculate the distance between the time series subsequence with the same variable (*e.g.*, the distance between the first variable (red time series on top) and S_1). Thus, the MST representation of a time series instance is a vector, as shown in the rightmost part. □

Algorithm 5.3 is the pseudo-code of multivariate shapelet transformation from final shapelet discovery (Lines 3−15), and shapelet transformation for the original dataset (Lines 17−23).

First, we do the clustering among all the new representations of the shapelet candidates $f(\Omega)$ using a standard clustering algorithm (Line 3). We use $(f(\Omega))^i$ to denote the ith cluster. Then, the nearest one to the centroid in each cluster is added to the set $f(\mathcal{S})$. In Lines 13−14, the utility of each candidate is used to select the top-k candidates. We trace $f(\mathcal{S})$ to the original dataset to retrieve their corresponding original subsequences (Line 15). They form the final shapelets $\mathcal{S}_k$ for the dataset, where $|\mathcal{S}_k| = k$.

Finally, the transformation computes the distance when the shapelet and the time series that has the same variable (Line 19). The distance between them is calculated by Equation 2.2 (Line 20). After the calculation between one instance in the original time series dataset and all the shapelets, the MST representation of the instance is denoted as $\tilde{\mathbb{T}}_m$ (Line 22). Then, the MST representation of the dataset is $\tilde{\mathbb{D}} = \tilde{\mathbb{T}}^{M \times k}$.

Algorithm 5.3: Multivariate Shapelet Transformation

Input: MTS dataset $\mathbb{D} = \mathbb{T}^{M \times V \times N}$, $f(\Omega)$, k
Output: Shapelets $\mathcal{S}_k$

Initialize the priority queue $f(\mathcal{S}) = \varnothing, \mathcal{S} = \varnothing$;
{shapelet discovery}
$\bigcup_{i=1}^{Y} (f(\Omega))^i \leftarrow \mathsf{kmeans}(f(\Omega))$;
for $i = \{1, 2, \cdots, Y\}$ **do** `// each cluster`
    $min = +\infty$;
    $f(\mathcal{S}).\mathsf{push}(f(min))$;
    foreach $f(e) \in (f(\Omega))^i$ **do**
        $tmp = ||(f(\Omega))^i.\mathsf{centroid} - f(e)||_2^2$;
        if $tmp < min$ **then**
            $min = tmp$;
            $f(\mathcal{S}).\mathsf{pop}()$;
            $f(\mathcal{S}).\mathsf{push}(f(e))$;
Calculate $\mathcal{U}$ (Equation 8 in the full paper) for each candidate in $f(\mathcal{S})$;
Sort each candidate based on $\mathcal{U}$ and select top-k candidates, denoted as $f(\mathcal{S}_k)$;
Retrieve $\mathcal{S}_k$ of $f(\mathcal{S}_k)$ from $\mathbb{D}$;
{Shapelet transformation}
for $m = \{1, 2, \cdots, M\}$ **do**
    for $j = \{1, 2, \cdots, k\}$ **do**
        $v = S_j.variable$;
        $d_{m,j} = \mathsf{dist}(\mathbb{T}_m^v, S_j)$;
        $\tilde{\mathbb{T}}_m.\mathsf{append}(d_{m,j})$;
    $\tilde{\mathbb{T}}_m =< d_{m,1}, d_{m,2}, \cdots, d_{m,k} >$;
$\tilde{\mathbb{D}} = \tilde{\mathbb{T}}^{M \times k}$;
return $\mathcal{S}_k$

After MST, the dataset $\mathbb{D}$ is reduced from $M \times V \times N$ to $M \times k$, where $|\mathcal{S}_k| = k$ and k is significantly smaller than $V \times N$.

When the transformation of all the multivariate time series instances is completed, some standard classifiers (*e.g.*, SVM) can be exploited to learn a classification model from the transformed representation. In this thesis, we adopt SVM with a linear kernel so that we can observe the weights of the shapelets for classification.

5.3 Experimental Results

5.3.1 Environment

We have implemented the proposed method[1] in PYTHON. For sustainability, we put the source code on google drive [2]. All the experiments were conducted on a machine with two Xeon E5-2630v3 @ 2.4GHz (2S/8C) / 128GB RAM / 64 GB SWAP and two NVIDIA Tesla K80, running on CentOS 7.3 (64-bit).

5.3.2 Datasets and Parameters

A well-known benchmark of multivariate time series datasets, namely UEA ARCHIVE, was tested. The detailed information of the datasets can be obtained from [42][3]. Table 5.1 shows the datasets' information, settings of the experiments with the datasets, where *Train, Test, Dimension*, *Length*, and *Class* are the numbers of time series in the training set and the testing set, the variable of each instance, the length of time series, and the number of classes, respectively.

The following are some parameters used in our experiment. We follow the default hyperparameters of the network from [39]. The batch size, the number of channels, the kernel size of the convolutional network, and the network depth are set to 10, 40, 3, and 10, respectively. The learning rate is kept fixed at the low value of $\eta = 0.001$,

[1] To promote reproducibility, our source code is made public at https://bit.ly/2VxHrJY.

[2] https://drive.google.com/file/d/1R7-Qgr3RpjSnwID1fn-59bTl9lstieZJ/view?usp=sharing

[3] http://www.timeseriesclassification.com

Table 5.1: Multivariate time series datasets information from UEA ARCHIVE

Dataset	Train	Test	Dimension	Length	Class
ArticularyWordRecognition	275	300	9	144	25
AtrialFibrillation	15	15	2	640	3
BasicMotions	40	40	6	100	4
CharacterTrajectories	1422	1436	3	182	20
Cricket	108	72	6	1197	12
DuckDuckGeese	60	40	1345	270	5
EigenWorms	128	131	6	17984	5
Epilepsy	137	138	3	206	4
ERing	30	30	4	65	6
EthanolConcentration	261	263	3	1751	4
FaceDetection	5890	3524	144	62	2
FingerMovements	316	100	28	50	2
HandMovementDirection	320	147	10	400	4
Handwriting	150	850	3	152	26
Heartbeat	204	205	61	405	2
InsectWingbeat	30000	20000	200	78	10
JapaneseVowels	270	370	12	29	9
Libras	180	180	2	45	15
LSST	2459	2466	6	36	14
MotorImagery	278	100	64	3000	2
NATOPS	180	180	24	51	6
PEMS-SF	267	173	963	144	7
PenDigits	7494	3498	2	8	10
Phoneme	3315	3353	11	217	39
RacketSports	151	152	6	30	4
SelfRegulationSCP1	268	293	6	896	2
SelfRegulationSCP2	200	180	7	1152	2
SpokenArabicDigits	6599	2199	13	93	10
StandWalkJump	12	15	4	2500	3
UWaveGestureLibrary	120	320	3	315	8

while the number of epochs for network training is 400. μ in Equation 5.1 is set to 0.2, $\lambda = 1$ for the triplet loss function. The β in Equation 5.15 is 0.5.

5.3.3 Baselines

We compared ShapeNet with seven different methods. We provide only brief details of each. Interested readers may refer to the original paper for further information.

- **Three benchmarks [42].** Three benchmark classifiers (EDI, $DTWI$, and $DTWD$) are based on Euclidean Distance (EDI), dimension-independent dynamic time warping ($DTWI$), and dimension-dependent dynamic time warping ($DTWD$) [79].
- **MLSTM-FCNs [58].** MLSTM-FCNs is a deep learning framework trans-

forming the LSTM-FCN models of univariate time series into multivariate time series by augmenting it with squeeze-and-excitation block.

- **WEASEL-MUSE [56].** WEASEL-MUSE is a bag-of-pattern based approach with statistical feature selection, variable window lengths and SAX for multivariate time series classification.
- **Negative samples (NS) [60].** This method applies several negative samples when training their neural network, then SVM is utilized to do the final classification.
- **TapNet [61].** TapNet is a novel multivariate time series classification model with an attentional prototype network to harness the strengths of both traditional and deep learning based approaches.

5.3.4 Experiments on Convergence of Mdc-CNN

We verify the convergence of Mdc-CNN, which depends on the parameters from Section 5.3.2. For instance, the convergences of the learning algorithm on four datasets, AtrialFibrillation, Basicmotions, StandWalkJump, and UWaveGestureLibrary, are illustrated in Figure 5.9.

All the losses converge very smoothly as the training proceeds on all four datasets. We can also observe that the loss converges quickly at the beginning, and then stabilizes. Similar trends can be observed from the rest of the datasets. This verifies the effectiveness of our cluster-wise triplet loss.

5.3.5 Experiments on Accuracy

5.3.5.1 Comparison with other methods

The experimental accuracies of the baseline results are all taken from the original papers [42], [60] and [61], respectively. We only consider the normalized datasets for the experiment. The overall classification accuracy results for the datasets are presented in Table 5.2. The accuracy results of ShapeNet are the mean values of 10

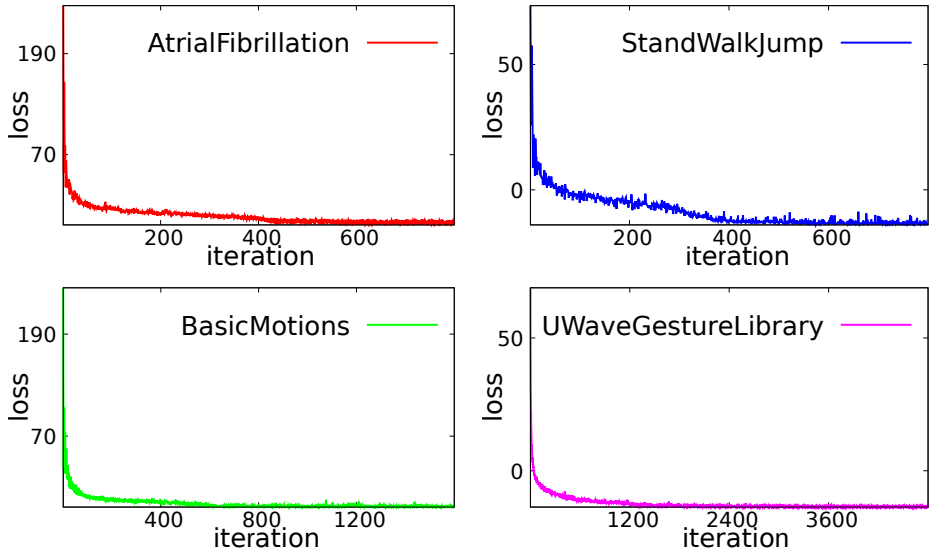


Figure 5.9: The convergences of the learning algorithm on some multivariate time series datasets

runs and the standard deviations of all the datasets are less than 0.01.

From Table 5.2, we can observe that the overall accuracy of ShapeNet is the best of all the methods compared. Moreover, ShapeNet performs best in 14 datasets, more than the other three benchmarked methods. The total best accuracy of ShapeNet is almost two times better than those of NS, TapNet, WEASEL-MUSE, and MLSTM-FCNs, and clearly even better than those of other methods. ShapeNet is clearly more accurate in some datasets, such as AtrialFibrillation and StandWalkJump. A probable reason is that high-quality shapelets do exist in those datasets and ShapeNet can discover them for classification. Our accuracies on 1-to-1-Losses datasets are only slightly lower than those of WEASEL-MUSE (*e.g.*, , Cricket, Epilepsy), NS (*e.g.*, JapaneseVowels, Libras) and TapNet (*e.g.*, PenDigits, SpokenArabicDigits).

5.3.5.2 Friedman test and Wilcoxon test

We follow the process described in [69] to conduct the Friedman test and Wilcoxon-signed rank test with Holm's α (5%) [71] for all the methods.

The Friedman test is a non-parametric statistical test to detect the differences in 30 datasets across eight methods. Our statistical significance is $p = 0.00$, which is

Table 5.2: Accuracy of ShapeNet and the related methods on UEA ARCHIVE

Dataset	EDI	DTWI	DTWD	MLSTM -FCNs	WEASEL +MUSE	NS	TapNet	ShapeNet
ArticularyWordRecognition	0.97	0.98	0.987	0.973	**0.99**	0.987	0.987	0.987
AtrialFibrillation	0.267	0.267	0.22	0.267	0.333	0.133	0.333	**0.4**
BasicMotions	0.676	**1**	0.975	0.95	**1**	**1**	**1**	**1**
CharacterTrajectories	0.964	0.969	0.989	0.985	0.99	0.994	**0.997**	0.98
Cricket	0.944	0.986	**1**	0.917	**1**	0.986	0.958	0.986
DuckDuckGeese	0.275	0.55	0.6	0.675	0.575	0.675	0.575	**0.725**
EigenWorms	0.549	N/A	0.618	0.504	**0.89**	0.878	0.489	0.878
Epilepsy	0.666	0.978	0.964	0.761	**1**	0.957	0.971	0.987
ERing	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**
EthanolConcentration	0.293	0.304	0.323	**0.373**	0.43	0.236	0.323	0.312
FaceDetection	0.519	N/A	0.529	0.545	0.545	0.528	0.556	**0.602**
FingerMovements	0.55	0.52	0.53	**0.58**	0.49	0.54	0.53	**0.58**
HandMovementDirection	0.278	0.306	0.231	0.365	0.365	0.27	**0.378**	0.338
Handwriting	0.2	0.316	0.286	0.286	**0.605**	0.533	0.357	0.451
Heartbeat	0.619	0.658	0.717	0.663	0.727	0.737	0.751	**0.756**
InsectWingbeat	0.128	N/A	N/A	0.167	N/A	0.16	0.208	**0.25**
JapaneseVowels	0.924	0.959	0.949	0.976	0.973	**0.989**	0.965	0.984
Libras	0.833	**0.894**	0.87	0.856	0.878	0.867	0.85	0.856
LSST	0.456	0.575	0.551	0.373	**0.59**	0.558	0.568	**0.59**
MotorImagery	0.51	N/A	0.5	0.51	0.5	0.54	0.59	**0.61**
NATOPS	0.85	0.85	0.883	0.889	0.87	**0.944**	0.939	0.883
PEMS-SF	0.705	0.734	0.711	0.699	N/A	0.688	**0.751**	**0.751**
PenDigits	0.973	0.939	0.977	0.978	0.948	**0.983**	0.98	0.977
Phoneme	0.104	0.151	0.151	0.11	0.19	0.246	0.175	**0.298**
RacketSports	0.868	0.842	0.803	0.803	**0.934**	0.862	0.868	0.882
SelfRegulationSCP1	0.771	0.765	0.775	**0.874**	0.71	0.846	0.652	0.782
SelfRegulationSCP2	0.483	0.533	0.539	0.472	0.46	0.556	0.55	**0.578**
SpokenArabicDigits	0.967	0.959	0.963	**0.99**	0.982	0.956	0.983	0.975
StandWalkJump	0.2	0.333	0.2	0.067	0.333	0.4	0.4	**0.533**
UWaveGestureLibrary	0.881	0.868	0.903	0.891	**0.916**	0.884	0.894	0.906
Total best acc	1	3	2	4	10	5	5	14
Ours 1-to-1-Wins	29	26	22	21	15	18	20	-
Ours 1-to-1-Draws	1	3	5	3	3	5	5	-
Ours 1-to-1-Losses	0	1	3	6	12	7	5	-
Rank Mean	6.2	5.43	4.77	4.6	3.47	3.67	3.23	2.23
Wilcoxon Test p-value	0.000	0.000	0.000	0.001	0.183	0.819	0.002	-

smaller than $\alpha = 0.05$. Thus, we reject the null hypothesis, and there is a significant difference among these eight methods.

We note that ShapeNet ranks the $1st$ on average among all the compared methods. We further conducted the Wilcoxon test against all baselines and found out that all results are statistically significant at $p < 0.05$, except WEASEL-MUSE, NS from the last row in Table 5.2.

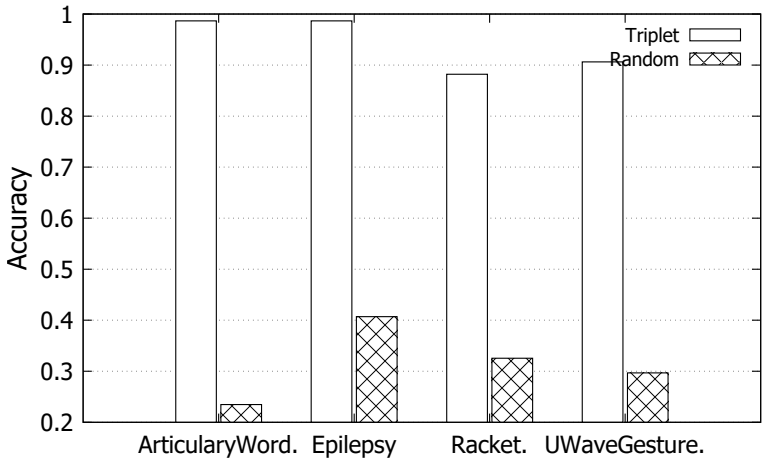


Figure 5.10: Triplet sampling vs random sampling of final shapelets

5.3.5.3 Triplet sampling vs. random sampling

To study the performance of our triplet sampling, we compare with random triplet sampling to train the network. We present the results from four multivariate time series datasets, namely ArticularyWordRecognition, Epilepsy, RacketSports and UWaveGestureLibrary, in Figure 5.10. Figure 5.10 shows the results of final accuracy: our triplet sampling is evidently the best of the four datasets.

5.3.5.4 Utility-based vs. random selection

To study the effectiveness of the utility function for selecting final shapelets in Section 5.2.5, we conduct an experiment to compare it with random selection. The clustering number is 200 and the value of k is 50. The random selection number is 50.

We report the final classification accuracy on four multivariate time series datasets, ArticularyWordRecognition, Epilepsy, RacketSports and UWaveGestureLibrary as examples. They are shown in Figure 5.11. The same trend can be found in other datasets. Among all four datasets, the accuracy of our utility-based method is clearly better than those of random selection, which shows its superiority ability to discover high-quality shapelets.

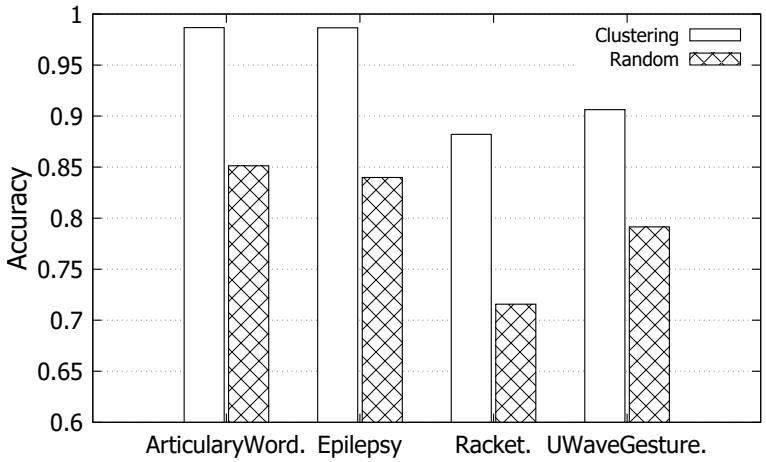


Figure 5.11: Utility-based vs random selection of final shapelets

5.3.5.5 Varying shapelet numbers

We compare the impact of different number of top-k shapelets from 200 clusters on the final accuracy of ShapeNet on four multivariate time series datasets: ArticularyWord., Epilepsy, RacketSports, and UWaveGestureLibrary.

Figure 5.12 shows accuracy by varying shapelet numbers. The accuracy increases rapidly as the number of shapelets increases from 5 to 50 in all four datasets, and then decreases slightly. This tendency is more evident in the ArticularyWordRecognition dataset than the other datasets since ArticularyWordRecognition has 25 classes. Thus, it is much harder to do the classification when the shapelet number is small (*e.g.*, 5). Based on this observation, the default shapelet number of all the datasets is set to 50 in Section 5.3.5.

5.3.6 Experiments on Interpretability

We further investigate the shapelets' interpretability, which is a strength of shapelet-based methods. We report two shapelets (*i.e.*, $k = 2$) generated by ShapeNet from two datasets. These datasets are chosen simply because they can be presented without much domain knowledge.

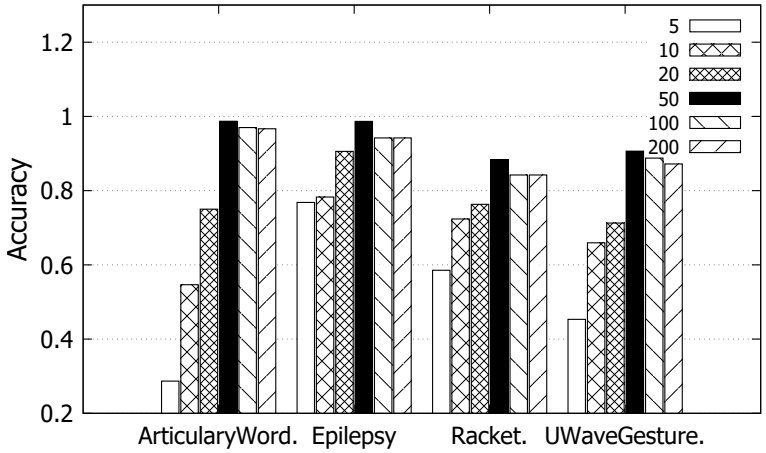


Figure 5.12: Multivariate time series classification accuracy by varying 6 shapelet numbers

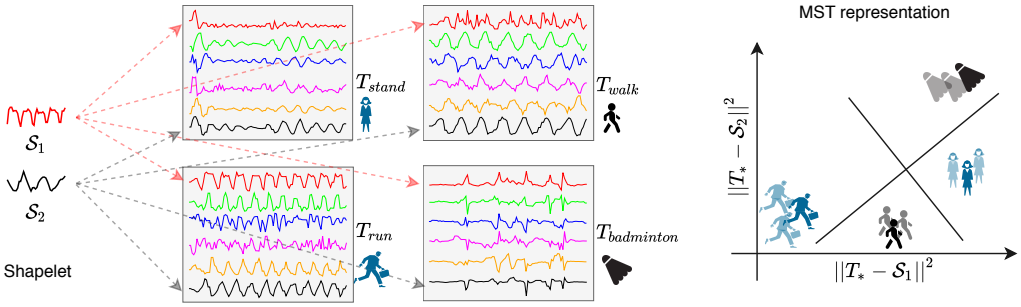


Figure 5.13: An example of multivariate shapelet transformation on Basicmotions [42]

5.3.6.1 Interpreting Basicmotions' shapelets

Two interesting shapelets, S_1 and S_2, are discovered from the Basicmotions dataset (leftmost plots) in Figure 5.13. S_1 describes the acceleration of the x-axis and S_2 depicts the angular velocity of the z-axis. The shapelets selected by ShapeNet are from the first and fifth variables, which shows the differing importance of the variables. The middle plots show four multivariate time series from four classes of the dataset. Different colors show different variables. The distance can only be calculated between the time series of the same variable (visually of the same color). The distances to two shapelets project the multivariate time series into a two-dimensional space (rightmost plot). Then, the transformed representations are

classified by a linear classifier. The result shows that S_2 is effective in distinguishing the badminton motion from others. S_1 can distinguish walking and running from others. Finally, *both* S_1 and S_2 can identify standing from others.

We note that the MST representation w.r.t shapelets is easier to interpret than the raw data and some knowledge can be observed. For example, standing and badminton are similar w.r.t S_1, which is counter-intuitive. It turns out that when waiting for the badminton, many players just stand.

5.3.6.2 Interpreting AtrialFibrillation's shapelets

We use AtrialFibrillation, which is an ECG dataset with two variables, as an example, to show the interpretability of the discovered multivariate shapelets. There are three classes in the dataset, namely "non-termination atrial fibrillation", "self-terminating at least one minute", and "terminating immediately". They are labeled as N, S, and T, respectively.

From the brief description of AtrialFibrillation, we can know that the terminating time of the three classes is T $<$ S $<$ N. However, the raw data are hard to understand even in a plot form. In Figure 5.14, our shapelets, S_1 and S_2, transform all the original time series into two-dimensional space. In the MST representation, readers can easily follow the terminating time of each class. The larger the magnitude in the new space, the more time for terminating on the original time series.

5.4 Chapter Summary

This chapter has proposed a novel shapelet-neural network approach for multivariate time series classification, ShapeNet. We propose Mdc-CNN to learn time series subsequences of various lengths into unified space and propose a cluster-wise triplet loss to train the network in an unsupervised fashion. We adopt MST to obtain the MST representation of time series. After the transformation, we employ SVM with a linear kernel to do the classification. The experimental results show that

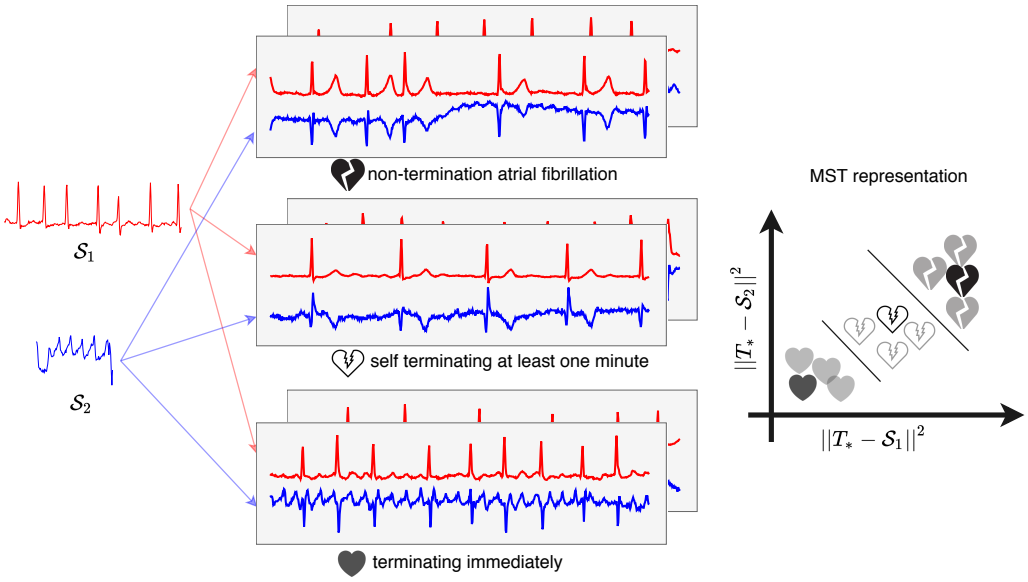


Figure 5.14: An example of multivariate shapelet transformation on Atrialfibrillation [42]

the classification accuracy of ShapeNet is superior to seven compared methods. The learning algorithm converges quickly, and the utility function is effective. The number of shapelets can be set to 50 (by default) for the highest accuracy. The interpretability of shapelets is illustrated with two case studies.

Chapter 6

Shapelet Autoencoder for Time Series Clustering

In this chapter, we investigate the problem of time series clustering. The studied problem and time series clustering preliminaries are introduced in Section 6.1. Section 6.2 proposes the details of AUTOSHAPE, which is an autoencoder network-based shapelet approach for both univariate and multivariate time series clustering, for discovering a set of high-quality shapelets under the unsupervised learning. Section 6.3 presents our experimental results. Finally, we summarize this chapter in Section 6.4.

6.1 Preliminaries and Problem Statement

In this section, we present some preliminaries and problem statement.

Definition 6.1. Unsupervised Shapelet S [14]. An unsupervised shapelet S of the length $|S|$ is a time series subsequence, which represents a group of time series instances and discriminate it from other groups, That is, for all T_j in this group, $\mathsf{dist}(T_j, S)$ is smaller than $\mathsf{dist}(T_k, S)$, where T_k is time series in other groups. □

Problem statement. Given a multivariate time series dataset $\mathbb{D}$[1], consisting of M multivariate time series instances $\mathbb{T}_1, \mathbb{T}_2, \cdots, \mathbb{T}_M$ with V variables, this thesis

[1]An univariate time series dataset D is a special case of an MTS dataset $\mathbb{D}$.

investigates an unsupervised shapelet discovery method for multivariate time series clustering. □

6.2 Autoencoder for Shapelets (AutoShape)

In this section, we propose an autoencoder-based shapelet approach, namely AutoShape. As the name suggests, AutoShape adopts the autoencoder approach. It learns the unified embeddings for shapelet discovery and meanwhile maintains original time series subsequences' shapes to understand the raw dataset. More specifically, we exploit an autoencoder network to learn the general unified embeddings of time series subsequences (shapelet candidates) using four objectives, namely ① self-supervised loss, ② diversity loss, ③ reconstruction loss and ④ Davies–Bouldin index (DBI) objective. We *jointly learn* the shapelets with all the four objectives without using labels.

6.2.1 Shapelet Discovery

We first present shapelet candidate generation and then introduce the objectives designed to train an autoencoder in detail.

6.2.1.1 Shapelet candidate generation

We apply sliding windows of different sizes to generate abundant shapelet candidates from the original multivariate time series[2]. A variate label is then annotated to each candidate for shapelet transformation in Section 6.2.2.3. The shapelet candidate generation procedure is summarized in Algorithm 6.1.

6.2.1.2 Self-supervised loss

We aim to learn a unified embedding of variable-length shapelet candidates of different variables. We adopt a cluster-wise triplet loss [43] as the self-supervised loss to learn the embedding in an unsupervised manner. The cluster-wise triplet loss

[2]Unless otherwise specified, we use the MTS as an example to explain in the following section.

Algorithm 6.1: Shapelet candidate generation

Input: MTS dataset $\mathbb{D} = \mathbb{T}^{M \times V \times N}$, sliding window size ϕ
Output: Shapelet candidates Ω

for l *in* ϕ **do**
  Initialize $\Omega_l = \varnothing$;
  for $m = \{1, 2, \cdots, M\}$ **do**
    for $v = \{1, 2, \cdots, V\}$ **do**
      for $i = \{1, 2, \cdots, N - l + 1\}$ **do**
        $e = T^v(i, i + l - 1)$; // subsequence
        $\Omega_l = \Omega_l + e$;
return Ω

function is defined with (i) the distance between the anchor and multiple positive samples ($\mathcal{D}_{AP}$), (ii) that between the anchor and multiple negative samples ($\mathcal{D}_{AN}$), and (iii) the sum of intra distance $\mathcal{D}_{intra}$ among positives and negatives, respectively. For self-containedness, we recall the cluster-wise triplet loss below.

$$\mathcal{L}_{Triplet}\left(f_e(x), f_e(\boldsymbol{x}^+), f_e(\boldsymbol{x}^-)\right) = \log \frac{\mathcal{D}_{AP} + \mu}{\mathcal{D}_{AN}} + \beta \mathcal{D}_{intra}, \tag{6.1}$$

where x is the anchor, $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ denote the set of positive and negative samples, respectively. μ denotes a margin, and β is a hyperparameter.

The encoder network $f_e(\cdot)$ maps from an original time series space to a latent space. The embedding function is $f_e : x \rightarrow h$. The function is learned using the self-supervised loss. According to [43], the encoder network can be parameterized by any neural network architecture of choice, with the only requirement that they obey causal ordering (*i.e.*, no future value impacts the current value). Here, we implement the encoder network using a Temporal Convolutional Network (TCN) [39].

6.2.1.3 Diversity loss

We propose our diversity loss for the autoencoder to discover shapelets of high quality.

Shapelet candidates are initially variable-length time series subsequences of different variables. AUTOSHAPE embeds all the shapelet candidates from the original

space into a new common space through the cluster-wise triplet loss. Following the protocol of selecting shapelets with diversity in previous research, namely USSL [13] and DTCR [63], we select the diverse shapelets for the next step, namely, shapelet transformation. Then, we cluster the representations of shapelet candidates in the new common space. After clustering, several clusters of representations are generated. We select the candidate, which is the closest to each centroid in each cluster. We penalize the following loss (Equation 6.2), which considers both the size of each cluster and the distances among all the selected candidates to achieve discrimination.

$$\mathcal{L}_{Diversity} = e^{-\sum_{i=1}^{Y}(\log(i.size)+\log\sum_{j=1}^{Y}||f_e(x_i)-f_e(x_j)||_2^2)} \tag{6.2}$$

where $f_e(x_i)$ is the closest representation to the centroid of cluster i, and $f_e(\cdot)$ is the encoder network. The first part of the exponent denotes the size of the ith cluster. The second part is the distance between the representation of the ith cluster and the representations of other clusters.

The diversity loss is designed for selecting shapelets with two characteristics. The cluster size of the representation determines the universality of the candidate and the distance shows the heterogeneity of the clusters.

6.2.1.4 Reconstruction loss

We next present a decoder network $f_d(\cdot)$, similar to the previous work on IDEC [45], DTC, [62] and DTCR [63]. The decoder network is guided by the MSE loss [80]. The representation $h = f_e(x)$ is the input for the decoder to reconstruct the original subsequence under the MSE loss. The recovery function is $f_d : h \rightarrow \tilde{x}$, where $\tilde{x}$ is the decoded time series subsequence.

$$\mathcal{L}_{Reconstruction} = ||x - \tilde{x}||_2^2 \tag{6.3}$$

6.2.2 Shapelet Adjustment

After selecting the shapelets, we use them to transform the original time series into a transformed representation, where each element is the Euclidean distance between the original time series and one of the shapelets.

Specifically, shapelet transformation [16] is a method to transform a time series T_j, w.r.t. the shapelets $\mathcal{S} : \{S_1, \cdots, S_k\}$, into a new data space (d_1^j,$\cdots$,d_k^j), where $d_i^j = \mathsf{dist}(T_j^v, S_i)$, and $\mathsf{dist}(T_j^v, S_i)$ is the distance between T_j^v and a shapelet S_i in $\mathcal{S}$. The distance of sequence T_p of the length n_p and T_q of the length n_q is denoted as (w.l.o.g. assuming $n_q \geq n_p$),

$$\mathsf{dist}(T_p, T_q) = \min_{j=1,\cdots,n_q-n_p+1} \frac{1}{n_p} \sum_{l=1}^{n_p} (t_q^{j+l-1} - t_p^l)^2.$$

6.2.2.1 DBI loss

We apply a classical clustering method (*e.g.*, K-means) on the transformed representations and then propose a DBI objective to inform adjustment of the shapelets. DBI is chosen because it does not need the ground truth for measuring, which is consistent with the unsupervised learning of AutoShape.

$$\mathcal{L}_{DBI} = \frac{1}{C_{num}} \sum_{i=1}^{C_{num}} \max_{j \neq i} \frac{A_i + A_j}{M_{i,j}}, \tag{6.4}$$

where C_{num} is the cluster number, A_i is the average distance between each element of cluster i and the centroid of that cluster, a.k.a., cluster diameter. $M_{i,j}$ is the distance between cluster centroids i and j.

In order to calculate the derivative of the loss function, all the involved functions of the model need to be differentiable. However, the maximum function of Equation 6.4 is not continuous and differentiable. We, therefore, introduce a differentiable approximation to the maximum function [76]. For the sake of organizational clarity, we simplify $R_{i,j} = \frac{A_i + A_j}{M_{i,j}}$. A differentiable approximation of Equation 6.4 is stated below.

$$\mathcal{L}_{DBI} \approx \tilde{\mathcal{L}}_{DBI} = \frac{1}{C_{num}} \sum_{i=1}^{C_{num}} \frac{\sum_{j=1}^{N} R_{i,j} \cdot e^{\alpha \cdot R_{i,j}}}{\sum_{j=1}^{N} e^{\alpha \cdot R_{i,j}}},$$

when $\alpha \to +\infty$, $\tilde{\mathcal{L}}_{DBI}$ approaches the true maximum $\mathcal{L}_{DBI}$.

An autoencoder network is applied to jointly learn the time series subsequences representations to select high-quality shapelets for the transformation. Our shapelets for transformation are decoded by the autoencoder, not necessarily restricted to the raw time series. At the same time, the reconstruction loss of AUTOSHAPE maintains the shapes of raw time series subsequences in the final shapelets, rather than learning some subsequences very different from the original time series.

6.2.2.2 Overall loss function

Finally, the overall loss $\mathcal{L}_{AS}$ of AUTOSHAPE is defined by:

$$\mathcal{L}_{AS} = \mathcal{L}_{Reconstruction} + \lambda \mathcal{L}_{Triplet} + \mathcal{L}_{Diversity} + \mathcal{L}_{DBI}, \tag{6.5}$$

where λ is the regularization parameter.

The overall loss (Equation 6.5) is minimized to jointly learn the shapelets for the transformation (as illustrated with Figure 1.8). After shapelet candidate generation, ① $\mathcal{L}_{Triplet}$ learns latent representations $\mathbf{Z}$ for time series subsequences to capture their characteristics. ② $\mathcal{L}_{Diversity}$ trains the shapelets with both universality and heterogeneity. ③ $\mathcal{L}_{Reconstruction}$ makes the reconstruction of the latent representations to preserve the shapes of the shapelets. Then, the selected shapelets (black and green ones) are exploited to transform the original time series (orange color) into the transformed representations (using vectors to represent the transformed representations). Then, the clustering algorithm (K-means) is applied to the transformed representations. ④ $\mathcal{L}_{DBI}$ is computed from the clustering results and used to adjust the shapelets, to improve the final clustering performance. All the loss functions model the encoder network, but the reconstruction loss and the DBI loss build the

decoder network only.

6.2.2.3 Overall algorithm

Algorithm 6.2 first shows the details of determining final shapelets under the learning process (Lines 2-13) and then the shapelet transformation process (Lines 15-21).

We initialize the final shapelet set (Line 2) and parameters of the network (Line 3) at the beginning. Our learning approach iterates for each epoch (Line 4) and each batch (Line 5) to update the representations and network parameters. The new representations $\mathcal{H}$ are learned through the encoder network $f_e(\cdot)$ (Line 6) and then decoded by $f_d(\cdot)$ (Line 7). We do the clustering on the transformed representation for $\mathcal{L}_{DBI}$ (Line 8). The overall loss $\mathcal{L}_{AS}$ is calculated (Line 9). The representations and weights are updated in the negative direction of the stochastic gradient of the loss (Lines 10-11). We sort all the embeddings of the candidates from Ω under the $\mathcal{L}_{Diversity}$ (Line 12). The top-k decoded embeddings of the candidates are selected for final shapelets (Line 13).

Finally, we explain the shapelet transformation procedure with the multivariate time series dataset. We compute the distance between the shapelet and the time series instance having the same variable (Line 17). The distance between them is calculated by using the formula at the beginning of Section 6.2.2 (Line 18). After the calculation between one time series instance from the original time series dataset and all the shapelets (Line 16), the transformed representation of the instance is denoted as $\tilde{\mathbb{T}}_m$ (Lines 19-20). Then, the representation of the original dataset $\mathbb{D}$ is transformed to $\tilde{\mathbb{D}} = \tilde{\mathbb{T}}^{M \times k}$ (Line 21).

Complexity analysis. After the transformation, the multivariate time series dataset $\mathbb{D}$ is reduced from $M \times V \times N$ to $M \times k$, where $|\mathcal{S}_k| = k$ and k is significantly smaller than $V \times N$. When the transformation of all the time series instances is done, some standard clustering methods (*e.g.*, K-means) can be exploited to do the clustering from the transformed representation. The time complexity of Lines 2-13 and Lines

Algorithm 6.2: Determining final shapelets

Input: MTS dataset $\mathbb{D} = \mathbb{T}^{M \times V \times N}$, candidates set Ω, learning rate η, epoch number $\mathcal{E}$, final shapelet number k

Output: Final shapelets $\mathcal{S}_k$

1 {shapelet discovery}
2 Initialize $\mathcal{S} = \varnothing$;
3 Initialize parameters of the autoencoder network $\mathcal{W}$;
4 **for** *epoch* $= 1 \rightarrow \mathcal{E}$ **do**
5 **for** $\mathcal{X} \in \Omega$ **do** // Candidates from Algo.5.1
6 $\mathcal{H} \leftarrow f_e(\mathcal{X})$; // $\mathcal{L}_{Triplet}, \mathcal{L}_{Diversity}$
7 $\tilde{\mathcal{X}} \leftarrow f_d(\mathcal{H})$; // $\mathcal{L}_{Reconstruction}$
8 kmeans$(ST(\tilde{\mathcal{X}}))$; // $\mathcal{L}_{DBI}$
9 $\mathcal{L}_{AS} \leftarrow \mathcal{L}_{AS}(\mathcal{X})$;
10 $\Delta\mathcal{W} \leftarrow -\eta \frac{\partial \mathcal{L}}{\partial \mathcal{W}}$;
11 $\mathcal{W} \leftarrow \mathcal{W} + \Delta\mathcal{W}$;
12 $f_e(\Omega)$.sort$(\mathcal{L}_{Diversity})$;
13 $\mathcal{S}_k = f_d(f_e(\Omega))$.pop$(k)$;
14 {Shapelet transformation}
15 **for** $m = \{1, 2, \cdots, M\}$ **do**
16 **for** $j = \{1, 2, \cdots, k\}$ **do**
17 $v = S_j.variable$;
18 $d_{m,j} = \text{dist}(\mathbb{T}_m^v, S_j)$;
19 $\tilde{\mathbb{T}}_m$.append$(d_{m,j})$;
20 $\tilde{\mathbb{T}}_m =< d_{m,1}, d_{m,2}, \cdots, d_{m,k} >$;
21 $\tilde{\mathbb{D}} = \tilde{\mathbb{T}}^{M \times k}$;
22 **return** $\mathcal{S}_k$

15-21 are $O(\frac{|MVN|}{|\mathcal{X}|} \cdot \mathcal{E} + MVN \cdot \log(MVN) + k)$ and $O(Mk)$, respectively. Hence, the time complexity of Algorithm 6.2 is $O(\frac{|MVN|}{|\mathcal{X}|} \cdot \mathcal{E})$. Therefore, it is not surprising that the network learning part is the bottleneck of the time complexity.

6.3 Experimental Results

In this section, we first present the comprehensive experiments conducted AutoShape with 14 related methods on the UCR datasets. The methods compared on UCR (univariate) datasets with AutoShape are the same in DTCR [63]. Thus, we ignore the details of each method in this thesis. Interested readers may refer to the original papers. We then report the results of AutoShape on the UEA (multivariate)

datasets with 5 related methods particularly in Section 6.3.4.

6.3.1 Experimental Settings

We have implemented the proposed method in PYTHON. All the experiments were conducted on a machine with two Xeon E5-2630v3 @ 2.4GHz (2S/8C) / 128GB RAM / 64 GB SWAP and two NVIDIA Tesla K80, running on CentOS 7.3 (64-bit).

We follow the default hyperparameters of the network from [39]. The following are some important parameters used in our experiment. The batch size, the number of channels, the kernel size of the convolutional network, and the network depth are set to 10, 40, 3, and 10, respectively. The learning rate is kept fixed at the small value of $\eta = 0.001$, while the number of epochs for network training is 400. λ is set to 0.01 for the overall loss function Equation 6.5. The number of shapelets is chosen from $\{1, 2, 5, 10, 20\}$. We try various lengths of sliding windows (namely, the lengths of shapelet candidates) ranging from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. Each number means a ratio of the original time series' length (*e.g.*, 0.1 means 10% of the original time series' length).

6.3.2 Baselines

We find that USSL [13] and DTCR [63] can significantly outperform some existing methods like, UDFS [81], KSC [82], k-DBA [83], NDFS [84], RUFS [85], RSFS [86], and some deep representation learning methods, DEC [44], IDEC [45], DTC [62]. We compare the following methods with AUTOSHAPE.

- **K-means [87].** Using K-means on the entire original time series.
- **UDFS [81].** Unsupervised discriminative feature selection with $l_{2,1}$-norm regularized.
- **NDFS [84].** Non-negative discriminative feature selection with non-negative spectral analysis.
- **RUFS [85].** Robust unsupervised discriminative feature selection with

robust orthogonal non-negative matrix factorization.

- **RSFS [86].** Robust spectral learning for unsupervised feature selection with sparse graph embedding.
- **KSC [82].** A pairwise scaling distance for K-means clustering and spectral norm for centroid computation.
- **KDBA [83].** Dynamic time warpping barycenter averaging (DBA) for K-means clustering.
- **k-Shape [64].** Adopting a scalable iterative refinement procedure to explore the shapes under a normalized cross-correlation measure to calculate the distance of two time series.
- **U-shapelet [11].** Discovering shapelets without label for time series clustering.
- **USSL [13].** Learning salient subsequences from unlabeled time series with shapelet regularization, spectral analysis and pseudo label.
- **DTC [62].** An autoencoder for temporal dimensionality reduction and a novel temporal clustering layer for cluster assignment, jointly optimizing the clustering and the dimensionality reduction.
- **DEC [44].** A method that simultaneously learns feature representations and cluster assignments using deep neural networks.
- **IDEC [45].** Manipulating feature space to scatter data points using a clustering loss as guidance, to constrain the manipulation and maintain the local structure of data generating distribution with an autoencoder.
- **DTCR [63].** Learning cluster-specific hidden temporal representations with temporal reconstruction loss, K-means objective, and classification task.

6.3.3 Experiments on Univariate Time Series

We have followed the protocol used in the previous work, k-Shape [64], USSL [13] and DTCR [63]. Thirty-six datasets from a well-known benchmark of time series

datasets, namely UCR archive, were tested. Detailed information on the datasets can be obtained from [17].

We take normalized mutual information (NMI) [13] as the metric for evaluating the methods. We omit the rand index (RI) results because we observe similar trends. The NMI close to 1 indicates high quality clustering [11]. The results of AutoShape are the mean values of 10 runs and the standard deviations of all the datasets are less than 0.005.

6.3.3.1 NMI on univariate time series

All the NMI results of the baselines are taken from the original papers [13] and [63], respectively. The overall NMI results for the 36 UCR datasets are presented in Table 6.1.

From Table 6.1, we can observe that the overall performance of AutoShape ranks first among the 15 compared methods. Moreover, AutoShape performs the best in 15 datasets, which is more than the other methods. The total highest NMI number of AutoShape is almost 1.7x larger than that of USSL and DTCR, and clearly more than that of other methods. AutoShape achieves much higher NMI numbers in some datasets, such as BirdChicken and ToeSegmentation1. Our results on 1-to-1-Losses datasets are only slightly lower than those of USSL (*e.g.*, Ham, Lighting2) and DTCR (*e.g.*, Car, ECGFiveDays).

6.3.3.2 Friedman test and Wilcoxon test.

We follow the process described in [69] to do the Friedman test and the Wilcoxon-signed rank test with Holm's α (5%) [71] for all the methods.

The Friedman test is a non-parametric statistical test to detect the differences in 36 datasets across 15 methods. Our statistical significance is $p = 0.00$, which is smaller than $\alpha = 0.05$. Thus, we reject the null hypothesis, and there is a significant difference among all 15 methods.

We then conduct the post-hoc analysis among all the methods. The results are

Table 6.1: Normalized mutual information (NMI) comparison on the UCR ARCHIVE

Dataset	K-means	UDFS	NDFS	RUFS	RSFS	KSC	KDBA	k-Shape	U-shapelet	DTC	USSL	DEC	IDEC	DTCR	Ours
Arrow	0.4816	0.524	0.4997	0.5975	0.5104	0.524	0.4816	0.524	0.3522	0.5	**0.6322**	0.31	0.2949	0.5513	0.5624
Beef	0.2925	0.2718	0.3647	0.3799	0.3597	0.3828	0.334	0.3338	0.3413	0.2751	0.3338	0.2463	0.2463	**0.5473**	0.3799
BeetleFly	0.0073	0.0371	0.1264	0.1919	0.2795	0.2215	0.2783	0.3456	0.5105	0.3456	0.531	0.0308	0.0082	**0.761**	0.531
BirdChicken	0.0371	0.0371	0.3988	0.1187	0.3002	0.3988	0.2167	0.3456	0.2783	0.0073	0.619	0.016	0.0082	0.531	**0.6352**
Car	0.254	0.2319	0.2361	0.2511	0.292	0.2719	0.2691	0.3771	0.3655	0.1892	0.465	0.2766	0.2972	**0.5021**	0.497
chlorineConcentration	0.0129	0.0138	0.0075	**0.0254**	0.0159	0.0147	0.0164	0	0.0135	0.0013	0.0133	0.0009	0.0008	0.0195	0.0133
coffee	0.5246	0.6945	**1**	0.2513	**1**	**1**	0.0778	**1**	**1**	0.5523	**1**	0.012	0.1431	0.6277	**1**
diatomsizeReduction	0.93	0.93	0.93	0.8734	0.8761	**1**	0.93	**1**	0.4849	0.6863	**1**	0.803	0.514	0.9418	**1**
phal.outl.agegroup	0.188	0.3262	0.1943	0.2762	0.3548	0.3331	0.4261	0.2911	0.2577	0.3406	0.3846	0.4405	0.44	**0.4553**	0.44
phal.outl.correct	0.0278	0.0473	0.0567	0.1071	0.0782	0.0261	0.0199	0.0527	0.0063	0.0115	0.1026	0.0011	0.015	0.118	**0.1333**
ECG200	0.1403	0.1854	0.1403	0.2668	0.2918	0.1403	0.1886	0.3682	0.1323	0.0918	0.3776	0.1885	0.2225	0.3691	**0.3928**
ECGFiveDays	0.0002	0.06	0.1296	0.0352	0.176	0.0682	0.1983	0.0002	0.1498	0.0022	0.6502	0.0178	0.0223	**0.8056**	0.7835
GunPoint	0.0126	0.022	0.0334	0.2405	0.0152	0.0126	0.1288	0.3653	0.3653	0.0194	0.4878	0.002	0.0031	**0.42**	0.4027
Ham	0.0093	0.0389	0.0595	0.098	0.0256	0.0595	0.0265	0.0517	0.0619	0.1016	**0.3411**	0.1508	0.1285	0.0989	0.3211
Herring	0.0013	0.0253	0.0225	0.0518	0.0236	0.0027	0	0.0027	0.1324	0.0143	0.1718	0.0306	0.0207	**0.2248**	0.2019
Lighting2	0.0038	0.0047	0.0851	0.1426	0.0326	0.1979	0.085	0.267	0.0144	0.1435	**0.3727**	0.06	0.1248	0.2289	0.353
Meat	0.251	0.2832	0.2416	0.1943	0.3016	0.2846	0.3661	0.2254	0.2716	0.225	0.9085	0.5176	0.225	**0.9653**	0.9437
phal.outl.agegroup	0.0219	0.1105	0.0416	0.1595	0.0968	0.1061	0.1148	0.0722	0.1491	0.139	0.278	0.2686	0.2199	**0.4661**	0.394
Mid.phal.outl.correct	0.0024	0.0713	0.015	0.0443	0.0321	0.0053	0.076	0.0349	0.0253	0.0079	0.2503	0.1005	0.0083	0.115	**0.2873**
Mid.phal.TW	0.4134	0.4276	0.4149	0.5366	0.4219	0.4486	0.4497	0.5229	0.4065	0.1156	0.9202	0.4509	0.3444	0.5503	**0.945**
MoteStrain	0.0551	0.1187	0.1919	0.1264	0.2373	0.3002	0.097	0.2215	0.0082	0.0094	**0.531**	0.3867	0.3821	0.4094	0.4257
OSULeaf	0.0208	0.02	0.0352	0.0246	0.0463	0.0421	0.0327	0.0126	0.0203	0.2201	0.3353	0.2141	0.2412	0.2599	**0.4432**
Plane	0.8598	0.8046	0.8414	0.8675	0.8736	0.9218	0.8784	0.9642	**1**	0.8678	**1**	0.8947	0.8947	0.9296	0.9982
phal.outl.ageGroup	0.0635	0.0182	0.083	0.0726	0.0938	0.0682	0.0377	0.011	0.0332	0.4153	0.6813	0.25	0.5396	0.5581	**0.693**
Prox.phal.TW	0.0082	0.0308	0.2215	0.1187	0.0809	0.1919	0.2167	0.1577	0.0107	0.6199	**1**	0.5864	0.3289	0.6539	0.8947
SonyAIBORobotSurface	0.6112	0.6122	0.6112	0.6278	0.6368	0.6129	0.5516	**0.7107**	0.5803	0.2559	0.5597	0.2773	0.4451	0.6634	0.6096
SonyAIBORobotSurfaceII	0.5444	0.4802	0.5413	0.5107	0.5406	0.5619	0.5481	0.011	0.5903	0.4257	0.6858	0.2214	0.2327	0.6121	**0.702**
SwedishLeaf	0.0168	0.0082	0.0934	0.0457	0.0269	0.0073	0.1277	0.1041	0.3456	0.6187	0.9186	0.5569	0.5573	0.6663	**0.934**
Symbols	0.778	0.7277	0.7593	0.7174	0.8027	0.8264	**0.9388**	0.6366	0.8691	0.7995	0.8821	0.7421	0.7419	0.8989	0.9147
ToeSegmentation1	0.0022	0.0089	0.2141	0.088	0.0174	0.0202	0.2712	0.3073	0.3073	0.0188	0.3351	0.001	0.001	0.3115	**0.461**
ToeSegmentation2	0.0863	0.0727	0.1713	0.1713	0.1625	0.0863	0.2627	0.0863	0.1519	0.0096	0.4308	0.0065	0.0118	0.3249	**0.4664**
TwoPatterns	0.4696	0.3393	0.4351	0.4678	0.4608	0.4705	0.4419	0.3949	0.2979	0.0119	0.4911	0.0195	0.0142	0.4713	**0.515**
TwoLeadECG	0	0.0004	0.1353	0.1238	0.0829	0.0011	0.0103	0	0.0529	0.0036	0.5471	0.0017	0.001	0.4614	**0.5654**
wafer	0.001	0.001	**0.0546**	0.0746	0.0194	0.001	0	0.001	0.001	0.0008	0.0492	0.0165	0.0188	0.0228	0.052
Wine	0.0031	0.0045	0.0259	0.0065	0.0096	0.0094	0.0211	0.0119	0.0171	0	**0.7511**	0.0018	0.1708	0.258	0.6045
WordsSynonyms	0.5435	0.4745	0.5396	0.5623	**0.5462**	0.4874	0.4527	0.4154	0.3933	0.3498	0.4984	0.4134	0.4387	0.5448	0.5112
Total best acc	0	0	2	1	2	2	1	3	2	0	9	0	0	9	15
Ours 1-to-1 Wins	34	34	32	31	33	31	34	33	33	36	24	35	35	23	-
Ours 1-to-1 Draws	0	0	1	1	1	2	0	2	1	0	4	0	1	0	-
Ours 1-to-1 Losses	2	2	3	4	2	3	2	1	2	0	8	1	0	13	-

visualized by the critical difference diagram in Figure 6.1. A thick horizontal line groups a set of methods that are not significantly different. We note that AUTOSHAPE clearly outperforms other approaches significantly except DTCR and USSL. However, AUTOSHAPE compared with DTCR, it is more capable of returning shapelets for interpretation of the dataset. The reconstruction loss of AUTOSHAPE maintains more details of final shapelets for interpretability, rather than learning some subsequences not in the original time series.

6.3.3.3 Varying shapelet numbers

We compare the impact of different shapelet numbers on the final NMI of AUTOSHAPE on four datasets, BirdChicken, Coffee, SwedishLeaf, and ToeSegmentation1.

Figure 6.2 shows the NMI by varying the shapelet numbers. Four different datasets show the different trends, which guide the appropriate shapelet number of the dataset. For example, the NMI stabilizes in BirdChicken through all the different numbers of shapelet. Thus, the shapelet number of BirdChicken is 1. The NMI rises

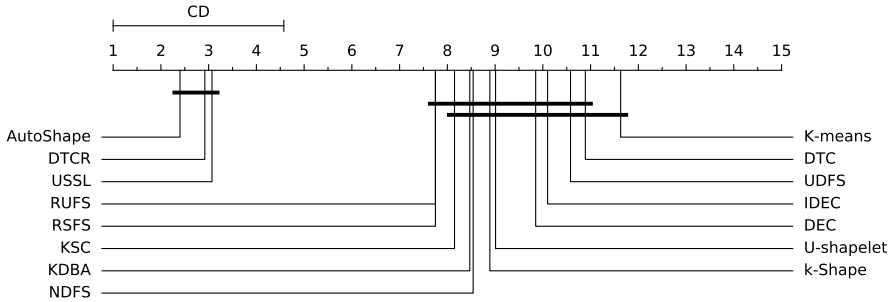


Figure 6.1: Critical difference diagram of the pairwise statistical comparison of 15 methods on the UCR Archive

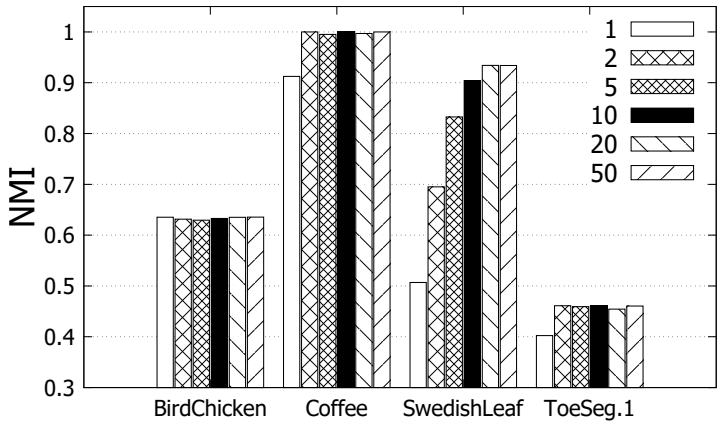


Figure 6.2: NMI by varying 5 numbers of shapelet on four UCR datasets [17]

rapidly with the increase in the shapelet number from 1 to 20 on SwedishLeaf, and then remains stable. Therefore, its shapelet number is set to 20.

6.3.3.4 Ablation study

We conduct an ablation study on AUTOSHAPE to further verify the effectiveness of $\mathcal{L}_{Triplet}$, $\mathcal{L}_{Diversity}$, and $\mathcal{L}_{DBI}$. We show a comparison between the full AUTOSHAPE and its three ablation models: AUTOSHAPE without self-supervised loss (w/o triplet), AUTOSHAPE without diversity loss (w/o diversity) and AUTOSHAPE without DBI loss (w/o DBI).

We observe in Table 6.2 that all three elements make important contributions to improving the final clustering performance. The general unified representation learning (self-supervised loss) plays a particularly important role, since the NMI

Table 6.2: Normalized mutual information (NMI) ablation comparisons on UCR ARCHIVE

Dataset	w/o triplet	w/o diversity	w/o DBI	AUTOSHAPE	Dataset	w/o triplet	w/o diversity	w/o DBI	AUTOSHAPE
Arrow	0.3942	0.4257	0.4344	**0.5624**	Mid.phal.outl.correct	0.2314	0.2534	0.2615	**0.2873**
Beef	0.3345	0.359	0.367	**0.3799**	Mid.phal.TW	0.8925	0.926	0.9307	**0.945**
BeetleFly	0.4894	0.5125	0.5019	**0.531**	MoteStrain	0.3748	0.4011	0.3924	**0.4257**
BirdChicken	0.541	0.6124	0.6099	**0.6352**	OSULeaf	0.3847	0.4127	0.4009	**0.4432**
Car	0.352	0.4347	0.4438	**0.497**	Plane	0.9045	0.985	0.972	**0.9982**
chlorineConcentration	0.0095	0.0107	0.0103	**0.0133**	phal.outl.ageGroup	0.6051	0.6567	0.6438	**0.693**
coffee	0.954	0.9854	0.9799	**1**	Prox.phal.TW	0.799	0.883	0.867	**0.8947**
diatomsizeReduction	0.9158	0.9867	0.9943	**1**	SonyAIBORobotSurface	0.5048	0.5568	0.5607	**0.6096**
phal.outl.agegroup	0.375	0.4069	0.4168	**0.44**	SonyAIBORobotSurfaceII	0.6287	0.6566	0.6601	**0.702**
phal.outl.correct	0.099	0.1154	0.1106	**0.1333**	SwedishLeaf	0.843	0.9247	0.9055	**0.934**
ECG200	0.3048	0.3457	0.3541	**0.3928**	Symbols	0.8253	0.8851	0.893	**0.9147**
ECGFiveDays	0.714	0.756	0.769	**0.7835**	ToeSegmentation1	0.416	0.431	0.4301	**0.461**
GunPoint	0.3201	0.3652	0.3549	**0.4027**	ToeSegmentation2	0.4057	0.4342	0.449	**0.4664**
Ham	0.2563	0.294	0.305	**0.3211**	TwoPatterns	0.4765	0.5017	0.5102	**0.515**
Herring	0.1347	0.1625	0.1599	**0.2019**	TwoLeadECG	0.4994	0.534	0.5276	**0.5654**
Lighting2	0.308	0.3241	0.3197	**0.353**	wafer	0.0257	0.0407	0.0398	**0.052**
Meat	0.882	0.9277	0.9189	**0.9437**	Wine	0.562	0.596	0.587	**0.6045**
phal.outl.agegroup	0.321	0.3467	0.3577	**0.394**	WordsSynonyms	0.468	0.497	0.502	**0.5112**

result of w/o triplet is always worse than the other two losses. In addition, we find that the shapelet candidates selection (diversity loss) and DBI objective clearly and consistently improve the final performance.

6.3.3.5 Experiments on UTS interpretability

We further investigate the interpretability of the shapelets, which is a strength of shapelet-based methods. We report the shapelets (k =1, 2) generated by AUTOSHAPE from three datasets. These datasets are chosen simply because they can be presented without much domain knowledge. From Figures 6.3, 6.4 and 6.6, we can observe that some subsequences of the original time series of the clusters are similar to their shapelets, which is the reason why they are clustered.

Case study 1: ToeSegmentation1. The ToeSegmentation1 dataset[3] is the left toe of z-axis value of human gait recognition (first used in [15]) derived from CMU Graphics Lab Motion Capture Database(CMU). The dataset consists of two categories, "normal walk" and "abnormal walk" containing hobble walk, hurt leg walk and others. In the abnormal walk category, the actors are pretending to have difficulty walking normally.

The interesting shapelets S_1 and S_2 are discovered from the ToeSegmentation1

[3] http://www.timeseriesclassification.com/description.php?Dataset=ToeSegmentation1

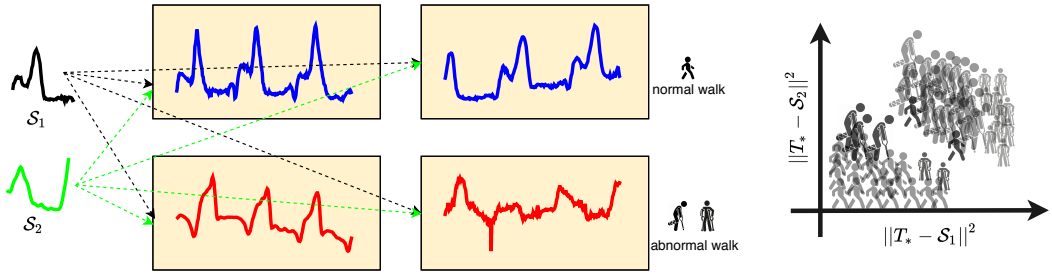


Figure 6.3: An example of shapelet transformation on ToeSegmentation1 [17]

dataset (leftmost plots) by AutoShape in Figure 6.3. The middle plots show four time series instances from two classes of the dataset. Different colors show different classes, blue for normal walk and red for abnormal walk. The distances to the shapelets project the original time series into a 2-dimensional space (rightmost plot). Then, the transformed representations are clustered by K-means. We can easily find that S_1 and S_2 are more frequent in normal walk class. S_1 shows one unit of normal walk and S_2 presents the interval of two consecutive walk units.

Case study 2: ItalyPowerDemand. ItalyPowerDemand[4] was derived from 12 monthly electrical power consumption time series from Italy in 1997. Two classes are in the dataset, summer from April to September, winter from October to March.

S_1 is learned by AutoShape showed in the left part of Figure 6.4. The original time series from summer class (blue) and winter class (red) are displayed in the middle. We can cluster the transformed representations into two classes effectively. From the learned shapelet, we can identify that the power demand in summer is lower than that in winter from 5am to 11pm. This is because the heating in the morning used during the winter time and air conditioning in summer was still fairly rare in Italy when the data were collected.

Case study 3: BirdChicken. We use BirdChicken[5] to show the effectiveness of the

[4] http://www.timeseriesclassification.com/description.php?Dataset=ItalyPowerDemand

[5] http://www.timeseriesclassification.com/description.php?Dataset=BirdChicken

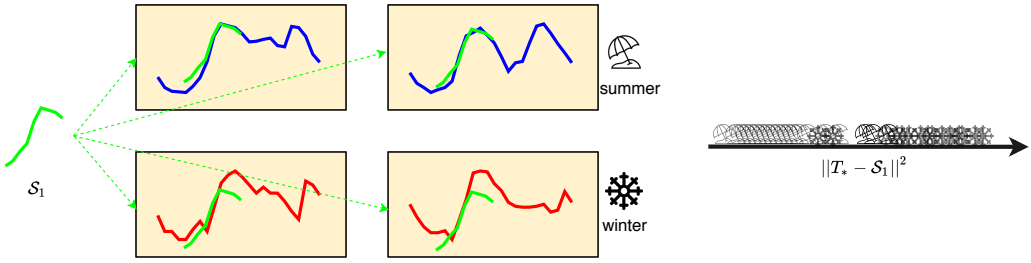


Figure 6.4: An example of shapelet transformation on ItalyPowerDemand [17]

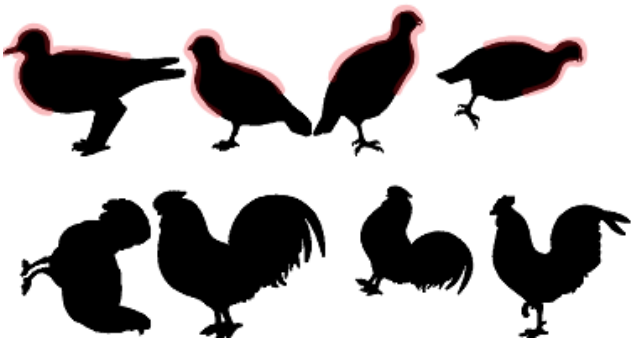


Figure 6.5: The original BirdChicken dataset [17], the red lines show the shapelet S_1 corresponding area

discovered unsupervised shapelets. BirdChicken contains binary images of bird and chicken, mapping into time series of distances to the center. There are two classes in the dataset, "bird" and "chicken". Figure 6.5 shows the original images.

In Figure 6.6, the shapelet S_1, transforms all the original time series into a 1-dimensional space. The left part shows the shapelets S_1 learned from the original time series. The middle shows the original time series from two classes, blue from bird and red from chicken. The right part plots the transformed representations of the original time series. Most of the predicted labels match the original labels. Shapelet S_1 is farther from chickens than from birds. We use red shadow lines to draw S_1 back onto the original figures of birds in Figure 6.5. It can be observed that the neck of a bird is thinner than its head. However, the neck of a chicken is thicker than its head.

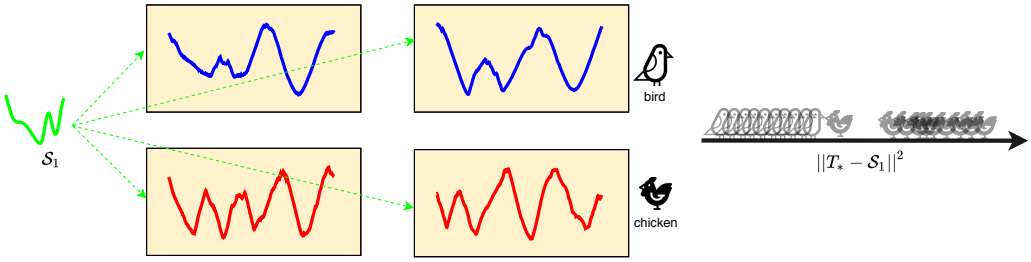


Figure 6.6: An example of shapelet transformation on BirdChicken [17]

6.3.4 Experiments on Multivariate Time Series

Next, we highlight the major results from the experiments conducted on multivariate time series. We take NMI [13] as the metric for evaluating the methods on multivariate time series. We omit the rand index (RI) results, as they exhibit similar trends. We choose K-means, GMM, k-Shape, USSL and DTCR as the compared methods[6]. Since they do not consider multivariate time series datasets, we simply concatenate the different variables into one. The results of all 6 methods are the mean values of 10 runs and the standard deviations of all the datasets are less than 0.01.

6.3.4.1 NMI on multivariate time series

The overall NMI results for the 30 UEA datasets are presented in Table 6.3.

From Table 6.3, we can observe that the overall performance of AutoShape ranks first among the 6 compared methods. Moreover, AutoShape performs the best in 24 multivariate time series datasets, which is more than the other 5 methods. One possible reason is that the compared methods are not specific designed to multivariate time series and AutoShape can learn the shapelets with high quality from different variables.

6.3.4.2 Friedman test and Wilcoxon test.

We follow the process described in [69] to do the Friedman test and the Wilcoxon-signed rank test with Holm's α (5%) [71] for all the methods.

[6]We obtained their source codes to test the performance on the UEA MTS datasets.

Table 6.3: Normalized mutual information (NMI) comparison on the UEA ARCHIVE

Dataset	K-means	GMM	k-Shape	USSL	DTCR	AUTOSHAPE
ArticularyWordRecognition	0.913298	0.915163	0.904142	0.4	0.8258	**0.9334**
AtrialFibrillation	6.06E-16	6.06E-16	0.325901	0.4347	0.2273	**0.4975**
BasicMotions	0.047414	6.41E-16	**0.940419**	0.4016	0.5012	0.5525
CharacterTrajectories	0.722946	0.506257	**0.794076**	0.4956	0.5214	0.7046
Cricket	0.799321	0.87574	**0.955548**	0.7944	0.8354	0.8451
DuckDuckGeese	0.042992	0.0395	0.2472	0.3124	0.3207	**0.3516**
EigenWorms	0.048604	0.0421	0.2152	0.3542	0.3557	**0.4208**
Epilepsy	0.295975	0.254485	0.370185	0.3945	0.2021	**0.4161**
ERing	0.351151	0.329892	0.304313	0.3045	0.363	**0.3735**
EthanolConcentration	0.023167	0.01159	0.010133	0.0968	0.054	**0.1346**
FaceDetection	0.000134	0.000404	9.35E-07	0.0573	0.0497	**0.1781**
FingerMovements	0.014989	0.000454	0.006442	0.0197	0.014	**0.0395**
HandMovementDirection	0.015407	0.041991	0.096969	0.0975	0.0771	**0.1823**
Handwriting	0.237603	0.225189	0.478558	0.4375	0.4627	**0.5087**
Heartbeat	0.005125	9.39E-16	0.036456	0.0532	0.0692	**0.0956**
InsectWingbeat	0.01283	0.026648	0.0573	0.0875	0.0957	**0.0992**
JapaneseVowels	0.720674	0.630336	**0.75363**	0.6695	0.4683	0.7153
Libras	0.62009	0.487687	0.619106	0.6057	0.6304	**0.6625**
LSST	0.029498	0	0.219803	0.2042	0.2578	**0.2843**
MotorImagery	0.009409	0.005467	0.010041	0.0176	0.0752	**0.1024**
NATOPS	0.567433	0.549475	0.57374	0.5454	0.2773	**0.5775**
PEMS-SF	0.33117	0.3548	0.3636	0.4125	0.4548	**0.4254**
PenDigits	0.664152	0.749791	0.166659	0.6753	0.4648	**0.6894**
Phoneme	0.149709	0.083097	0.166659	0.2059	0.2346	**0.2441**
RacketSports	0.369649	0.150482	**0.512878**	0.5076	0.2708	0.4635
SelfRegulationSCP1	0.118707	2.56E-15	0.183319	0.5452	0.107	**0.5805**
SelfRegulationSCP2	0.002484	0	0.001434	0.5783	0.0004	**0.6003**
SpokenArabicDigits	0.703619	0.169305	**0.883885**	0.2937	0.2193	0.4523
StandWalkJump	6.06E-16	6.06E-16	0.176256	0.5109	0.2306	**0.5224**
UWaveGestureLibrary	0.683235	0.669697	0.552855	0.6924	0.2493	**0.7053**
Total best acc	0	0	6	0	0	24
Ours 1-to-1 Wins	30	30	24	29	29	-
Ours 1-to-1 Draws	0	0	0	0	0	-
Ours 1-to-1 Losses	0	0	6	1	1	-

The Friedman test is used to detect the differences in 30 UEA datasets across 6 methods. Our statistical significance is $p = 0.00$, which is smaller than $\alpha = 0.05$. Thus, we reject the null hypothesis, and there is a significant difference among all 6 methods.

We then conduct the post-hoc analysis among all the compared methods. The results are visualized by the critical difference diagram in Figure 6.7. A thick horizontal line groups a set of methods that are not significantly different. We can clearly note that AUTOSHAPE outperforms all other 5 methods significantly.

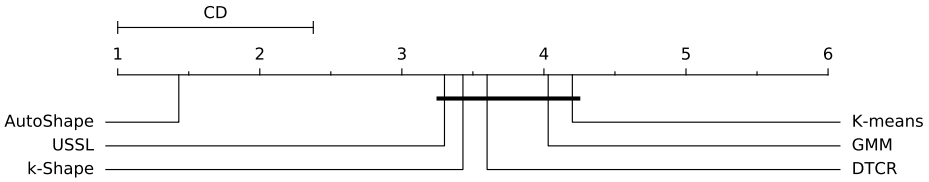


Figure 6.7: Critical difference diagram of the pairwise statistical comparison of 6 methods on the UEA Archive

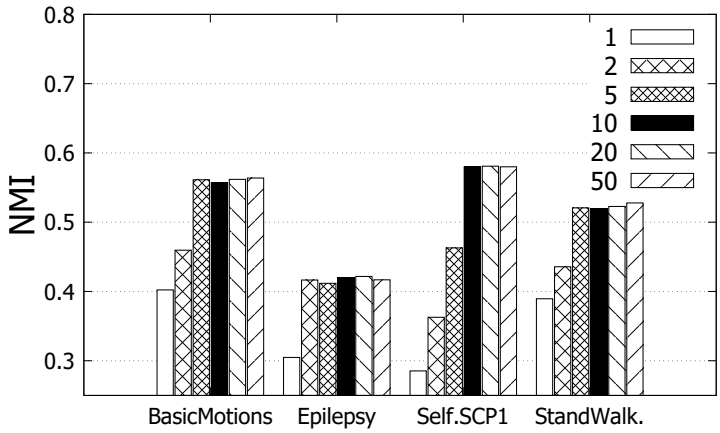


Figure 6.8: NMI by varying 5 numbers of shapelet on four UEA datasets [42]

6.3.4.3 Varying shapelet numbers

We further compare the impact of different shapelet numbers on the final NMI of AutoShape on four multivariate time series datasets, BasicMotions, Epilepsy, SelfRegulationSCP1, and StandWalkJump.

Figure 6.8 presents NMI by varying the 6 different shapelet numbers. Four different datasets exhibit different trends, which show the appropriate shapelet number selection for the dataset. For example, NMI increases rapidly with the increase in the shapelet number from 1 to 2 on Epilepsy, and then remains stable. Thus, its shapelet number is set to 2.

6.3.4.4 Experiments on MTS interpretability

Finally, we investigate the interpretability of the learned shapelets on multivariate time series. We describe the shapelets (*e.g.*, $k = 2$) generated by AutoShape from the Epilepsy dataset in Figures 6.9. Again, we choose the dataset simply because it can

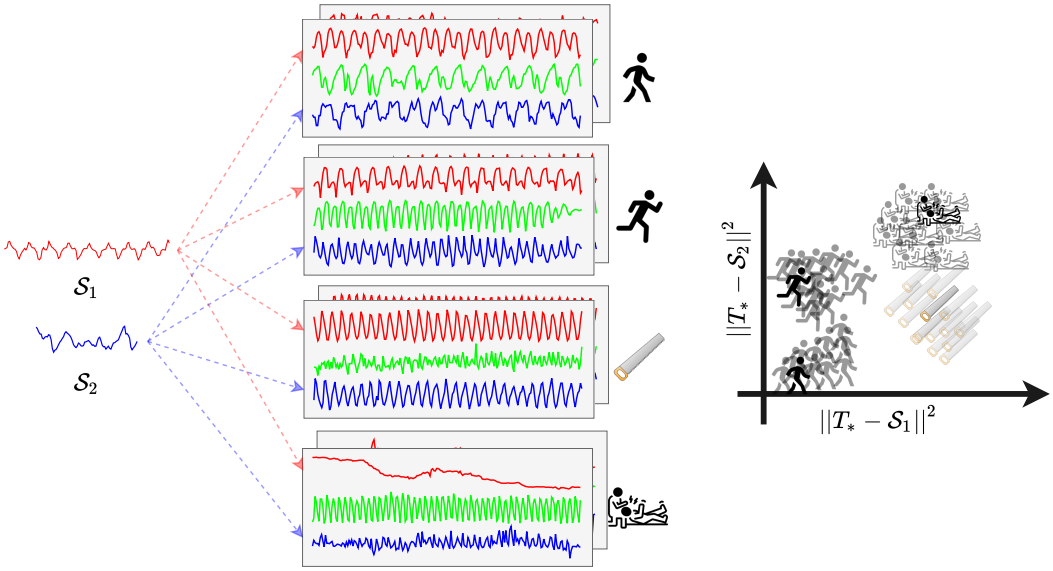


Figure 6.9: An example of shapelet transformation on one MTS dataset, called Epilepsy [42]

be illustrated without much domain knowledge. The Epilepsy dataset was generated with healthy participants simulating the class activities performed. The dataset consists of 4 categories, "walking", "running", "sawing" and "seizure mimicking".

The interesting shapelets S_1 and S_2 are learned from the Epilepsy dataset (leftmost plots) by AUTOSHAPE in Figure 6.9. The middle plots show the time series instances from four classes. Three different colors show the different variables. The original time series are projected into a 2-dimensional space (rightmost plot). We then employ the K-means to cluster the transformed representations. Most of the original time series can be smoothly clustered under the transformed representations.

6.4 Chapter Summary

This chapter has proposed a novel autoencoder-based shapelet approach for time series clustering, AUTOSHAPE. We propose an autoencoder network to learn the unified embeddings of shapelet candidates via the following objectives. Self-supervised loss is for learning the general embedding of subsequences. We propose diversity loss among shapelet candidates to select diverse candidates. The reconstruction loss

maintains the original time series' shapes for interpretability. DBI is an internal index to guide network learning for improving clustering performance. Extensive experiments show the superiority of our AutoShape to the other 14 and 5 compared methods on univariate time series and multivariate time series datasets, respectively. The interpretability of the learned shapelets is illustrated with three case studies on the UCR datasets and one case study on the UEA datasets.

Chapter 7

Conclusions

In this dissertation, we have tackled three time series analysis problems, namely time series classification, multivariate time series classification, and time series clustering based on our shapelet discovery.

7.1 Contributions

1. BSPCOVER **for univariate time series classification.** The first contribution of this dissertation is that we propose a novel efficient shapelet discovery method, called BSPCOVER, to discover a set of high-quality shapelet candidates for model building. Specifically, BSPCOVER generates abundant candidates via Symbolic Aggregate approXimation with sliding windowemploythe, then prunes identical and highly similar candidates via Bloom filters, and similarity matching, respectively. We next propose a p-Cover algorithm to efficiently determine discriminative shapelet candidates that maximally represent each time series class. Finally, any existing shapelet learning method can be adopted to build a classification model. We have conducted extensive experiments with well-known time series datasets and representative state-of-the-art methods. Results show that BSPCOVER speeds up the state-of-the-art methods by more than 70 times, and the accuracy is often comparable to or higher than existing works.

2. MPS **for univariate time series classification.** The fourth contribution of this

dissertation is that we propose to adopt a bagging technique for shapelet candidate generation, then prune shapelet candidates that violate the definition of shapelets, and measure the utility among shapelet candidates themselves and between raw time series. We have conducted comprehensive experiments on MPS with 10 competitive state-of-the-art methodsusing UCR Archive datasets. The run time of MPS is about 21 times faster than that of BSPCOVER (the latest method) on average. The accuracy results validate that MPS is comparable to or higher than existing work. Furthermore, we illustrate the shapelets' interpretability with a case study.

3. ShapeNet for multivariate time series classification. The second contribution of this dissertation is effective shapelet discovery for multivariate time series classification. In this dissertation, we propose a novel model, called ShapeNet, which embeds shapelet candidates of different lengths into a unified space for shapelet discovery. The network is trained using a cluster-wise triplet loss, which considers the distance between anchor and multiple positive (negative) samples and the distance between positive(negative) samples, which are important for convergence. We compute representative and diversified final shapelets rather than directly using all the embeddings for model building to avoid a large fraction of non-discriminative shapelet candidates. We have conducted experiments on ShapeNet with competitive state-of-the-art and benchmark methods using UEA MTS datasets. The results show that the accuracy of ShapeNet is the best of all the methods compared. Furthermore, we illustrate the shapelets' interpretability with two case studies.

4. AUTOSHAPE **for time series clustering.** The third contribution of this dissertation is that we propose a novel method called Autoencoder for Shapelets (AUTOSHAPE). We propose an autoencoder network to learn the representations of time series subsequences. For guiding the latent representation learning, we employ the latest self-supervised loss to learn the unified embeddings for variable-length shapelet candidates of different variables and propose the diversity loss to select the discriminating embeddings in the unified space. In addition, we propose the reconstruction loss

to recover shapelets in the original time series space to surface the features of the clusters. Finally, we adopt Davies–Bouldin index (DBI) to inform AutoShape of the clustering performance during learning. To evaluate the clustering performance on univariate time series (UTS), we have conducted experiments on AutoShape with 14 competitive state-of-the-art methods using UCR datasets. To study the performance of multivariate time series (MTS), we evaluate AutoShape on 30 UEAdatasets with 5 methods. The results validate that AutoShape is the best of all the methods compared. In four case studies, we interpret clusters with shapelets.

7.2 Future Directions

This dissertation aims to develop a shapelet-based approach for the time series analysis, such as classification and clustering. While the shapelet-based methods play an important role in time series analysis, our solutions still suffer from two major drawbacks. On the one hand, shapelets come from the raw time series data, which shrinks the existing scope of the shapelets. On the other hand, the current shapelet-based methods have only been tested in a few time series analysis, such as (multivariate) time series classification and clustering.

Accelerating Shapelet Discovery Process. Although the efficiency of the shapelet discovery has been improved by BSPCOVER and MPS significantly, the time series data grows rapidly nowadays and it is technically interesting to calculate the time series shapelets online to ensure a timely response. Thus, we could not only leverage high performance computing such as GPGPU to accelerate the computation but also plan to design a distributed shapelet discovery system to reduce the real-time processing. Finally, we could investigate strategies to balance the final accuracy and efficiency.

Enlarging Shapelet Discovery Scope. Since shapelets can be any time series (subsequences) of some length smaller than or equal to the length of the shortest time series. Thus there is an exponentially-many number of possible shapelet candidates.

Most of the shapelet-based methods assume that the shapelet candidates come from the raw time series dataset. This assumption may not always be necessary. Based on Chapter 5, we plan to utilize the scheme with generative adversarial network (GAN) to learn the high-quality shapelets.

Supporting More Time Series Analysis. We have two directions for supporting more time series applications. One is for the domain specific applications, *e.g.*, medicine, astronomy, and geoscience, which can utilize shapelet-based methods for the strength of accuracy, efficiency, and interpretability. The other is the application of time series, *e.g.*, anomaly detection, even forecasting. Based on the proposed ShapeNet Chapter 5 and AUTOSHAPE in Chapter 6 for time series clustering, we could further study the anomaly detection of time series. As for the time series forecasting based on shapelets, it is undoubtedly technically challenging and no related studies have been proposed nowadays. We plan to learn a shapelet network through the discovered shapelets for time series forecasting.

Appendix A

Learning the Shapelets

A.1 Overview

To make the prospectus self-contained, we illustrate the approach of utilizing the logistic regression to adjust the shapelet candidates, which transforms the SAX words Ω_C discovered from Chapter 3 back to the raw time series data. In this work, we build the traditional $|\mathcal{C}|$ one-vs-rest binary classifiers for time series datasets D with $|\mathcal{C}|$ labels. For each classifier, it has its own shapelets $\mathcal{S}_k$ to distinguish it from the others.

The rationale of this appendix is that the real shapelets which have the highest discrimination power may not occur in the training time series datasets. Thus, we try to utilize the learning logistic regression model with gradient descent to modify the shapelets for enhancing the accuracy of classification.

Similar to ELIS [1], BSPCOVER determines shapelets per class, whereas LTS [9] does not compute shapelets of each class separately. The advantages of the former approach are twofold. The first is on the interpretability. It is convenient to follow up the shapelets of each class. The second is that this class-oriented approach improves accuracies, *e.g.*, the accuracy results presented in Table 3.5.

A.2 Related Formula

We summarize the key steps of the learning time series shapelets, which can be fit to this work, as follows.

Soft-minimum function. Due to the non-differentiability of Eqn. 2.2, we replace the minimum distance function by a soft-minimum function, depicted in Eqn. A.1:

$$M_{soft}(T_j^i, \mathcal{S}_k) = \frac{\sum_{i=1}^{n-l_{\mathcal{S}_k}+1} D_{j,k,i} \mathrm{e}^{\alpha D_{j,k,i}}}{\sum_{i'=1}^{n-l_{\mathcal{S}_k}+1} \mathrm{e}^{\alpha D_{j,k,i'}}} \tag{A.1}$$

where α is a fixed number, equals to -25 and

$$D_{j,k,i} = \mathsf{dist}(T_j^i, \mathcal{S}_k) \tag{A.2}$$

Learning model. The model we use is a linear learning one which can predict target values $\hat{Y}$.

$$\hat{Y} = W_0 + \sum_{k=1}^{k_C} M_{soft}(T_j^i, \mathcal{S}_k) W_k, \quad \forall j \in \{1, \cdots, m\} \tag{A.3}$$

where W_0 and W_k are linear weights and k_C is the number of shapelets for class C.

Logistic sigmoid function. Here we adopt a Sigmoid function for prediction of target variable $\hat{Y}$ via a logistic regression loss.

$$\sigma(\hat{Y}) = \frac{1}{1 + e^{-\hat{Y}}} \tag{A.4}$$

Loss function. The loss function is denoted as:

$$\mathcal{L}(\hat{Y}, Y) = -Y \ln \sigma(\hat{Y}) - (1 - Y) \ln(1 - \sigma(\hat{Y})), \tag{A.5}$$

where Y is the true target value.

Objective function. The objective function for class C is therefore presented as follows:

$$\arg\min_{\mathcal{S},W} \mathcal{F} = \arg\min_{\mathcal{S},W} \sum_{j=1}^{m} \mathcal{L}(\hat{Y}_j, Y_j) + \lambda_W ||W||^2 \tag{A.6}$$

Algorithm A.1: Shapelet learning [1]

Input : Time series T, Learning Rate η, Iteration I, Regularization λ_W
Output : Shapelets $\mathcal{S}$ and Classification weights W for each class

Initializing $\mathcal{S}, W$ for each class;
{Learning Shapelets and Classification Weights}
for $iter = \{1, 2, \cdots, I\}$ **do**
  for $j = \{1, 2, \cdots, m\}$ **do**
    foreach $C \in \mathcal{C}$ **do**
      {Pre-compute soft-minimum function}
      for $s = \{1, 2, \cdots, k_C\}$ **do**
        $\xi[C][s] = M_{soft}(T_j, \mathcal{S}[C][s])$;
      end

      {Predict target value}
      $\hat{Y} = W[C][0]$;
      for $s = \{1, 2, \cdots, k_C\}$ **do**
        $\hat{Y} += \xi[C][s] * W[C][s]$
      end
      $\Sigma[C] = \sigma(\hat{Y})$;

      {Learning S and W}
      $W[C][0] -= \eta \frac{\partial \mathcal{F}_j}{\partial W[C][0]}$;
      for $s = \{1, 2, \cdots, k_C\}$ **do**
        $\mathcal{S}[C][s] -= \eta \frac{\partial \mathcal{F}_j}{\partial \mathcal{S}[C][s]}$;
        $W[C][s] -= \eta \frac{\partial \mathcal{F}_j}{\partial W[C][s]}$;
      end
    end
  end
end
return $\mathcal{S}, W$

where $\mathcal{S}$ is the set of shapelets for class C, W is the weight vector, λ_W is the regularization parameter, $\hat{Y}_j$ and Y_j are the predicted and ture target values of j_{th} time series for class C respectively. And the objective function $\mathcal{F}_j$ for per instance T_j is

$$\mathcal{F}_j = \mathcal{L}(\hat{Y}_j, Y_j) + \frac{\lambda_W}{m} \sum_{k=1}^{k_c} W_k^2 \tag{A.7}$$

The shapelet learning algorithm is presented in Algorithm A.1. The complexity is $O(Imn^2)$.

Bibliography

[1] Z. Fang, P. Wang, and W. Wang, "Efficient learning interpretable shapelets for accurate time series classification," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 2018, pp. 497–508.

[2] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, 2017.

[3] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *ACM SIGKDD international conference on Knowledge discovery and data mining workshop*, vol. 10, 1994, pp. 359–370.

[4] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering–a decade review," *Information Systems*, vol. 53, pp. 16–38, 2015.

[5] T. W. Liao, "Clustering of time series data—a survey," *Pattern recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[6] P.-Y. Zhou and K. C. Chan, "A model-based multivariate time series clustering algorithm," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2014, pp. 805–817.

[7] A. Singhal and D. E. Seborg, "Clustering multivariate time-series data," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 19, no. 8, pp. 427–438, 2005.

[8] H. Li, "Multivariate time series clustering based on common principal component analysis," *Neurocomputing*, vol. 349, pp. 239–247, 2019.

[9] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 392–401.

[10] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, 2013, pp. 668–676.

[11] J. Zakaria, A. Mueen, and E. Keogh, "Clustering time series using unsupervised-shapelets," in *2012 IEEE 12th International Conference on Data Mining*, 2012, pp. 785–794.

[12] L. Ulanova, N. Begum, and E. Keogh, "Scalable clustering of time series with u-shapelets," in *Proceedings of the 2015 SIAM international conference on data mining*, 2015, pp. 900–908.

[13] Q. Zhang, J. Wu, P. Zhang, G. Long, and C. Zhang, "Salient subsequence learning for time series clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2193–2207, 2018.

[14] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 947–956.

[15] L. Ye and E. Keogh, "Time series shapelets: A novel technique that allows accurate, interpretable and fast classification," *Data mining and knowledge discovery*, vol. 22, no. 1-2, pp. 149–182, 2011.

[16] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 289–297.

[17] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, *The ucr time series classification archive*, www.cs.ucr.edu/~eamonn/time_series_data/, Jul. 2015.

[18] T. Le Nguyen, S. Gsponer, and G. Ifrim, "Time series classification by sequence learning in all-subsequence space," in *2017 IEEE 33rd international conference on data engineering (ICDE)*, 2017, pp. 947–958.

[19] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[20] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003, pp. 2–11.

[21] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: The collective of transformation-based ensembles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, 2015.

[22] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: A review," *Data Mining and Knowledge Discovery*, pp. 1–47, 2019.

[23] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[24] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.

[25] J. Wang, Z. Wang, J. Li, and J. Wu, "Multilevel wavelet decomposition network for interpretable time series analysis," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2437–2446.

[26] M. Rhif, A. Ben Abbes, I. R. Farah, B. Martínez, and Y. Sang, "Wavelet transform application for/in non-stationary time-series analysis: A review," *Applied Sciences*, vol. 9, no. 7, p. 1345, 2019.

[27] G. Li, B. Choi, J. Xu, S. S. Bhowmick, K. P. Chun, and G. Wong, "Efficient shapelet discovery for time series classification," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[28] L. Hou, J. T. Kwok, and J. M. Zurada, "Efficient learning of timeseries shapelets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, pp. 1209–1215.

[29] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets," in *2016 IEEE International Conference on Data Mining*, 2016, pp. 1317–1322.

[30] C.-C. M. Yeh, N. Kavantzas, and E. Keogh, "Matrix profile iv: Using weakly labeled time series to predict outcomes," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1802–1812, 2017.

[31] H. A. Dau and E. Keogh, "Matrix profile v: A generic technique to incorporate domain knowledge into motif discovery," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 125–134.

[32] S. Alaee, K. Kamgar, and E. Keogh, "Matrix profile xxii: Exact discovery of time series motifs under dtw," *2020 IEEE 20th international conference on data mining (ICDM)*, 2020.

[33] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[34] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: A novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[35] A. Mueen, E. Keogh, and N. Young, "Logical-shapelets: An expressive primitive for time series classification," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1154–1162.

[36] Q. Ma, W. Zhuang, S. Li, D. Huang, and G. W. Cottrell, "Adversarial dynamic shapelet networks.," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 5069–5076.

[37] A. Bostrom and A. Bagnall, "A shapelet transform for multivariate time series classification," *arXiv preprint arXiv:1712.06428*, 2017.

[38] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme, "Fast classification of univariate and multivariate time series through shapelet discovery," *Knowledge and Information Systems*, vol. 49, no. 2, pp. 429–454, 2016.

[39] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[40] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[41] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[42] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The uea multivariate time series classification archive, 2018," *arXiv preprint arXiv:1811.00075*, 2018.

[43] G. Li, B. Choi, J. Xu, S. S. Bhowmick, K. P. Chun, and G. Wong, "Shapenet: A shapelet-neural network approach for multivariate time series classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[44] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[45] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation.," in *IJCAI international joint conference on artificial intelligence*, 2017, pp. 1753–1759.

[46] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 224–227, 1979.

[47] P. Schäfer and U. Leser, "Fast and accurate time series classification with weasel," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 637–646.

[48] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," *Journal of Intelligent Information Systems*, vol. 39, no. 2, pp. 287–315, 2012.

[49] P. Senin and S. Malinchik, "Sax-vsm: Interpretable time series classification using sax and vector space model," in *2013 IEEE 13th international conference on data mining*, 2013, pp. 1175–1180.

[50] G. Ifrim and C. Wiuf, "Bounded coordinate-descent for biological sequence classification in high dimensional predictor space," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 708–716.

[51] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh, "Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins," in *2016 IEEE 16th international conference on data mining (ICDM)*, IEEE, 2016, pp. 739–748.

[52] Y. He, X. Chu, and Y. Wang, "Neighbor profile: Bagging nearest neighbors for unsupervised time series mining," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, IEEE, 2020, pp. 373–384.

[53] J. Mei, M. Liu, Y.-F. Wang, and H. Gao, "Learning a mahalanobis distance-based dynamic time warping measure for multivariate time series classification," *IEEE Transactions on Cybernetics*, vol. 46, no. 6, pp. 1363–1374, 2016.

[54] M. G. Baydogan and G. Runger, "Learning a symbolic representation for multivariate time series classification," *Data Mining and Knowledge Discovery*, vol. 29, no. 2, pp. 400–422, 2015.

[55] H. Li, "Accurate and efficient classification based on common principal components analysis for multivariate time series," *Neurocomputing*, vol. 171, pp. 744–753, 2016.

[56] P. Schäfer and U. Leser, "Multivariate time series classification with weasel+ muse," *arXiv preprint arXiv:1711.11343*, 2017.

[57] S. Seto, W. Zhang, and Y. Zhou, "Multivariate time series classification using dynamic time warping template selection for human activity recognition," in *IEEE Symposium Series on Computational Intelligence*, 2015, pp. 1399–1406.

[58] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.

[59] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Frontiers of Computer Science*, vol. 10, no. 1, pp. 96–112, 2016.

[60] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[61] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, "Tapnet: Multivariate time series classification with attentional prototypical network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[62] N. S. Madiraju, "Deep temporal clustering: Fully unsupervised learning of time-domain features," PhD thesis, Arizona State University, 2018.

[63] Q. Ma, J. Zheng, S. Li, and G. W. Cottrell, "Learning representations for time series clustering," in *Advances in Neural Information Processing Systems*, 2019, pp. 3781–3791.

[64] J. Paparrizos and L. Gravano, "K-shape: Efficient and accurate clustering of time series," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015, pp. 1855–1870.

[65] R. J. Larsen, M. L. Marx, *et al.*, *An introduction to mathematical statistics and its applications*. Pearson, 2017, vol. 5.

[66] N. Begum and E. Keogh, "Rare time series motif discovery from unbounded streams," *Proceedings of the VLDB Endowment*, vol. 8, no. 2, pp. 149–160, 2014.

[67] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of operations research*, vol. 4, no. 3, pp. 233–235, 1979.

[68] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[69] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[70] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[71] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.

[72] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The handbook of brain theory and neural networks*, 1998, pp. 255–258.

[73] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, vol. 26, Curran Associates, Inc., 2013.

[74] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *Journal of Machine Learning Research*, vol. 11, no. Mar, pp. 1109–1135, 2010.

[75] Y. Goldberg and O. Levy, "Word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[76] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[77] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[78] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[79] M. Shokoohi-Yekta, J. Wang, and E. Keogh, "On the non-trivial generalization of dynamic time warping to the multi-dimensional case," in *Proceedings of the 2015 SIAM international conference on data mining*, SIAM, 2015, pp. 289–297.

[80] E. L. Lehmann and G. Casella, *Theory of point estimation*. Springer Science & Business Media, 2006.

[81] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, "L2, 1-norm regularized discriminative feature selection for unsupervised," in *IJCAI international joint conference on artificial intelligence*, 2011.

[82] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 177–186.

[83] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[84] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, "Unsupervised feature selection using nonnegative spectral analysis," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2012.

[85] M. Qian and C. Zhai, "Robust unsupervised feature selection," in *IJCAI international joint conference on artificial intelligence*, 2013.

[86] L. Shi, L. Du, and Y.-D. Shen, "Robust spectral learning for unsupervised feature selection," in *2014 IEEE International Conference on Data Mining (ICDM)*, 2014, pp. 977–982.

[87] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

List of Publications

- W.-K. Hui Vicki, **G. Li**, S. Li, C.-F. Yip Terry, Y.-K. Tse, W.-S. Wong Vincent, L.-H. Wong Grace, and K.-K. Choi Byron, "Novel shapelet patterns for time series classification (tsc) of afp and alt to predict hcc in patients with chronic hepatitis b on antiviral treatment," in *International Liver Congress (**ILC 2021**)*, 2021, Poster paper.
- **G. Li**, B. Choi, J. Xu, S. S. Bhowmick, K. P. Chun, and G. Wong, "Efficient shapelet discovery for time series classification (extended abstract)," in *The 37th IEEE International Conference on Data Engineering (**ICDE 2021**)*, 2021. DOI: 10.1109/ICDE51399.2021.00254, Poster paper.
- **G. Li**, B. Choi, j. Xu, S. S. Bhowmick, K. P. Chun, and G. Wong, "Shapenet: A shapelet-neural network approach for multivariate time series classification," in *The 35th AAAI Conference on Artificial Intelligence (**AAAI 2021**)*, 2021, Full paper.
- **G. Li**, B. Choi, S. S. Bhowmick, K. P. Chun, G. Wong, and S. Li, "Visualet: Visualizing shapelets for time series classification," in *29TH ACM International Conference on information and knowledge management (**CIKM 2020**)*, 2020, pp. 3429–3432. DOI: 10.1145/3340531.3417414, Demo paper.
- **G. Li**, B. Choi, J. Xu, S. S. Bhowmick, K. P. Chun, and G. Wong, "Efficient shapelet discovery for time series classification," *IEEE Transactions on Knowledge and Data Engineering (**TKDE**)*, 2020. DOI: 10.1109/TKDE.2020.2995870, Full paper.

- **G. Li**, N. Ng, P. Yi, Z. Zhang, and B. Choi, "Answering the why-not questions of graph query autocompletion," in *International Conference on Database Systems for Advanced Applications (**DASFAA 2018**)*, Springer, May 2018, pp. 332–341. DOI: 10.1007/978-3-319-91452-7_22, Short Paper.

CURRICULUM VITAE

Academic qualifications of the thesis author, Mr. Li Guozhong:

- Received the degree of Bachelor of Engineering from University of Electronic Science and Technology of China, July 2014.
- Received the degree of Master of Engineering from University of Electronic Science and Technology of China, July 2017.

August 2021