Novel Approaches Towards Fast and Accurate Time series Classification

BY

ANOOSHIRAVAN SHARABIANI

B.S., UNIVERSITY OF SCIENCE AND TECHNOLOGY, 1999

M.S., SHARIF UNIVERSITY OF TECHNOLOGY, 2001

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Industrial Engineering and Operations
Research in the Graduate College of the
University of Illinois at Chicago, 2018

Chicago, Illinois

Defense Committee:

Houshang Darabi, Chair and Advisor
Lev Reyzin, Mathematics, Statistics and Computer Science
Michael J. Scott, Mechanical and Industrial Engineering
Lin Li, Mechanical and Industrial Engineering
Mengqi Hu, Mechanical and Industrial Engineering

This thesis is dedicated to my wife Maryam Teimoori, my daughter Shailene and my parents, Nasrin and Behrooz Sharabiani, for their endless love, support and encouragement.

ACKNOWLEDGMENTS

I would like to seize this opportunity and thank several people without whom this work would never be possible.

I would like to thank my adviser Professor Houshang Darabi. I sincerely thank him for constantly supporting me and my family throughout my work at the University of Illinois at Chicago during last 5 years. He gave me the opportunity to succeed and believed in me throughout all the challenges in my work.

I also would like to thank my fellow lab mates at Process Mining and Intelligent System Analytics Team Lab, Samuel Harford, Hereford Johnson, Fazle Karim, Elnaz Douzali, Ilia Mokhtarian, Ashley Pimental, Angeline Kampert, Shun Chen and Ashkan Rezaei who provided a convivial environment for me to work and also helped me in my research.

I would like to express my gratitude to my committee members. Especially, I would like to thank Dr. Reyzin, Dr. Li, Dr. Scott, and Dr. Hu whose support and knowledge have been invaluable in my research.

I am thankful to my brothers Arash and Dr. Ashkan Sharabiani for their constant support and love in my life. Dr. Ashkan Sharabiani helped me a lot during my PhD program.

I am grateful to my wife, the love of my life, Maryam Teimoori. I am truly indebted for her love, her patience, and her support. I am also thankful to my parents Nasrin and Behrooz Sharabiani for their never-ending love, support, and guidance in life. They have always encouraged me to pursue this degree and be patient and resilient especially at difficult moments.

ASH

CONTRIBUTION OF AUTHORS

In Chapter 1, Chapter 2, Chapter 3 and Chapter 4, material from my own previously published work (A.Sharabiani, H.Darabi, A.Rezaei, S.Harford, H.Johnson and F. Karim, "Efficient Classification of Long Time Series by 3-Dimensional Dynamic Time Warping." IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47 Issue: 8. doi:10.1109/TSMC.2017.2699333), is reused. I was the primary author and major driver of this research. My research mentor, Dr. Houshang Daribi contributed to the writing of the manuscript. Ashkan Rezaei assisted me in transforming my code from R to C++. Sam Harford helped me in preparation of the figures. Hereford Johnson and Fazle Karim helped me by reviewing the manuscript before the submission.

Chapter 5 represents a manuscript (A.Sharabiani, H.Darabi, S.Harford, H.Johnson, F. Karim, E.Douzali and S.Chen, "Asymptotic Dynamic Time Warping Calculation with Utilizing Values Repetition", Knowledge and Information System Journal, 2017, In press) for which I was the primary author and major driver of the research. Dr. Houshang Daribi contributed to the writing of the manuscript. Sam Harford and Fazle Karim assisted me in coding the algorithms and performing the experiments. Elnaz Douzali helped me in preparation of the Latex file. Hereford Johnson and Shun Chen helped me by reviewing the manuscript before the submission.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBEREVAIATIONS

1-NN	First-Nearest-Neighbor
3D DTW	Three Dimensional Dynamic Time Warping
APCA	Adaptive Piecewise Constant Approximation
BOP	Bag-Of-Patterns
BOSS	Bag-Of-SFA-Symbols
BOSS VS	Bag-Of-SFA-Symbols in Vector Space
CCA	Control Chart Approximation
CHEBY	Chebyshev Polynomials
CID	Complexity Invariant Distance
COTE	Collective of Transformation Ensembles
DDDTW	Derivative Distance Dynamic Time Warping
DFT	Discrete Fourier Transform
DTDc	Derivative Transform Distance
DTW	Dynamic Time Warping
DTW CV	DTW with the warping window constraint set through Cross Validation
DWT	Discrete Wavelet Transform
EE	Elastic Ensemble
ED	Euclidean Distance
FS	Fast Shapelets
LB	Lower Bounding
LCSS	Longest Common Subsequence

LIST OF ABBEREVAIATIONS (continued)

LPS	Learned Pattern Similarity
LS	Learned Shapelet
MSM	Move-Split-Merge
PAA	Piecewise Aggregate Approximation
SAX	Symbolic Aggregate ApproXimation
SAX-VSM	Symbolic Aggregate ApproXimation Vector Space Model
SDA	Shape Description Alphabet
ST	Shapelete Transform
SVD	Singular Value Decomposition
SVM	Support Vector Machines
TSF	Time Series Forest
TSBF	Time Series Bag of Features
TWE	Time Wrap Edit
WDTW	Weighted Dynamic Time Warping
WDDTW	Weighted Derivative Dynamic Time Warping

SUMMARY

A time series is a collection of values made or recorded sequentially in time by live observations or sensors. Dynamic Time Warping (DTW) is an algorithm for measuring distance/similarity between two time series which may vary (i.e. warp) in timing. Throughout recent years, DTW has remained the most successful similarity measure in time series data mining (TSDM) because it is invariant to warping. DTW is applied in a variety of domains and problems. In particular, it has been successfully used in: robotics, bioacoustics, video games, computational photography, biometrics, medicine, music processing, bioinformatics, gesture recognition, metrology, image processing, seismology, entomology, anthropology, finance, etc.

First-Nearest-Neighbor Dynamic Time Warping (1-NN DTW) is the most widely used classification method on time series, and serves as a benchmark when compared to emerging techniques. Multiple recent studies show that for the problem of time series classification, 1-NN DTW is very hard to beat.

With the increasing demand for time series classification on low-resource devices, and the widespread sensory implemented technologies including wearables, the need for a fast and accurate time series classifier has never been higher. Although 1-NN DTW attains accurate results, it has a demanding cost in processing time due to its quadratic complexity in the length of time series. The state-of-the-art implementation of 1-NN DTW improves the running time by applying lower bounding, and early abandoning techniques. Nevertheless, it still turns out to be inefficient in classifying long time series.

The focus of this research is improving the efficiency of DTW while retaining its accuracy level on time series classification.

This thesis has three main contributions. The first and second contributions are the works that have been presented in the preliminary defense exam. The third will be presented in the final defense exam.

The first contribution of this thesis is that, it proposes a new approximation method for reducing the length of the time series as the input of DTW. This method is called Control Chart Approximation (CCA). CCA representation approximates raw time series by transforming them into a set of segments with aggregated values and durations forming a reduced 3-dimensional vector. The CCA transformation attempts to reduce noise, dimensionality, and storage of time series, while retaining the significant features of the original time series.

The second contribution of this thesis is extension of DTW in three dimension space as a distance measure for a 1-NN classifier. The method is called 1-Nearest Neighbor 3-Dimensional Dynamic Time Warping (1-NN 3D DTW). Our proposed methods, CCA and 1-NN 3D DTW, have focused on the reduction of data, while improving the processing time of 1-NN DTW on classification of long time series.

Lower bounding techniques and early abandoning method are developed to further reduce the time complexity in the 3-dimensional space. Using 85 time series benchmark datasets from the University of California, Riverside (UCR) archive - including 28 long-length (>500 points) time series datasets - show up to two orders of magnitude performance gain in running time compared to the state-of-the-art 1-NN DTW implementation while remaining competitive in terms of accuracy. Using 1-NN 3D DTW, we can obtain competitive tradeoffs between time and accuracy, and significantly increase classification efficiency (speed) especially on long-length time series.

The third contribution consists of developing Blocked Dynamic Time Warping (BDTW), a new similarity measure which works on run-length encoded time series. BDTW algorithm utilizes the repetition of values in time series to calculate the exact DTW for two-valued time series, and to calculate a close approximation of DTW for more-than-two-valued time series with repetition of any values.

By combining BDTW and Adaptive Piecewise Constant Approximation (APCA) method, a new DTW approximation method is proposed which works for all type of time series (with or without repetition of values).

BDTW Upper Bound is proposed as a new upper bound instead of the ED for pruning unhopeful warping alignments of time series with high value repetition rate. BDTW Lower Bound is also presented as a new lower bound for pruning unhopeful matches in similarity search of time series with high value repetition rate.

It has been shown that BDTW a) significantly reduces the processing time of DTW calculation for time series with high value repetition b) combined with APCA, serves as a DTW approximation method that beats traditional DTW approximation based on Piecewise Aggregate Approximation (PAA)-DTW-Projection approach in accuracy. c) is extendable to Constrained warping and Constrained BDTW processing time is significantly less than Constrained DTW processing time for time series with high value repetition.

The effectiveness of the BDTW and BDTW variations are shown on different applications, using the Almanac of Minutely Power dataset, the Refit Smart Homes dataset and 85 datasets in UCR time series classification archive.

1 INTRODUCTION

In this chapter, time series classification, the contributions and the outline of this thesis are presented. Material from my own previously published work [73], 2017 IEEE, is reused in this chapter.

1.1 Time Series Classification

Over the past few years, substantial amounts of data have been collected and analyzed to assist in optimal decision making. This collection and analysis of large data sets has become a primary contributor towards the innovation and growth of the current and emerging markets. A significant portion of this collected data is in the form of time series.

The exponentially increasing volume and complexity of time series is a result of emerging new sensing technologies (robot sensors, wearable sensors, smart meters, satellites, smart mobile phones, etc.), along with an influx of inexpensive storage. The key objective of time series analysis is to extract hidden insights from raw data. Time series classification is one of the most important tasks in time series analysis. There are varieties of time series classification techniques in the literature. Distance-based classification techniques such as 1-nearest neighbor require a similarity measure to calculate the distance (similarity) between two time series.

Euclidian distance (ED) [1] and dynamic time warping (DTW) [2]-[5] are the most popular distance-measures for time series. 1-nearest neighbor ED (1-NN ED) is often used for fast classification of the time series of equal length. However, its accuracy is highly sensitive to noise and it cannot be used when the lengths of time series are different.

1-nearest neighbor DTW (1-NN DTW) classifier is one of the best distance-based classifiers in terms of accuracy [6],[7]. It has been successfully applied in a variety of domains and problems such as analyzing robots sensory signals [8],[25], medicine [5], astronomy, biometric data, geology, historical manuscripts, speech/music, gesture, signature, and fingerprint recognition. 1-NN DTW can handle classification of time series of different length, and is typically used as a benchmark among time series classifiers [6]-[9]. The drawback of 1-NN DTW is its quadratic time complexity, which has reportedly undermined its usefulness in many use cases. Multiple research efforts would have used 1-NN DTW if it had not been too computationally expensive [9].

The state-of-the-art implementation of 1-NN DTW [9] reduces the computational complexity by omitting square root calculations, by using lower bounding, and by applying early abandoning. However, it is still inefficient when classifying long and large time series.

1.2 Contributions

In this effort, first a new representation method is propose, called Control Chart Approximation (CCA), which transforms raw time series data, from the 2-dimensional space of time-values, to a sequence of segments each forming a tuple of (segment begin time, segment average of values, segment duration); with the durations being z-normalized across the training time series. The data reduction results in reduced noise, space, and length of the time series before classification. An example of CCA transformation is presented in Figure 1.

To perform nearest neighbor classification in this new data space, the 3-dimensional dynamic time warping (1-NN 3D DTW) is proposed, which is based on 1-NN DTW and works in 3-dimensional space. We show that 1-NN 3D DTW significantly reduces the computation time of time series classification while still remaining competitive in terms of accuracy. The performance gain is much more significant on long time series, where the running time is orders of magnitude

faster than the state-of-the-art implementation of 1-NN DTW while scoring better or close in accuracy.

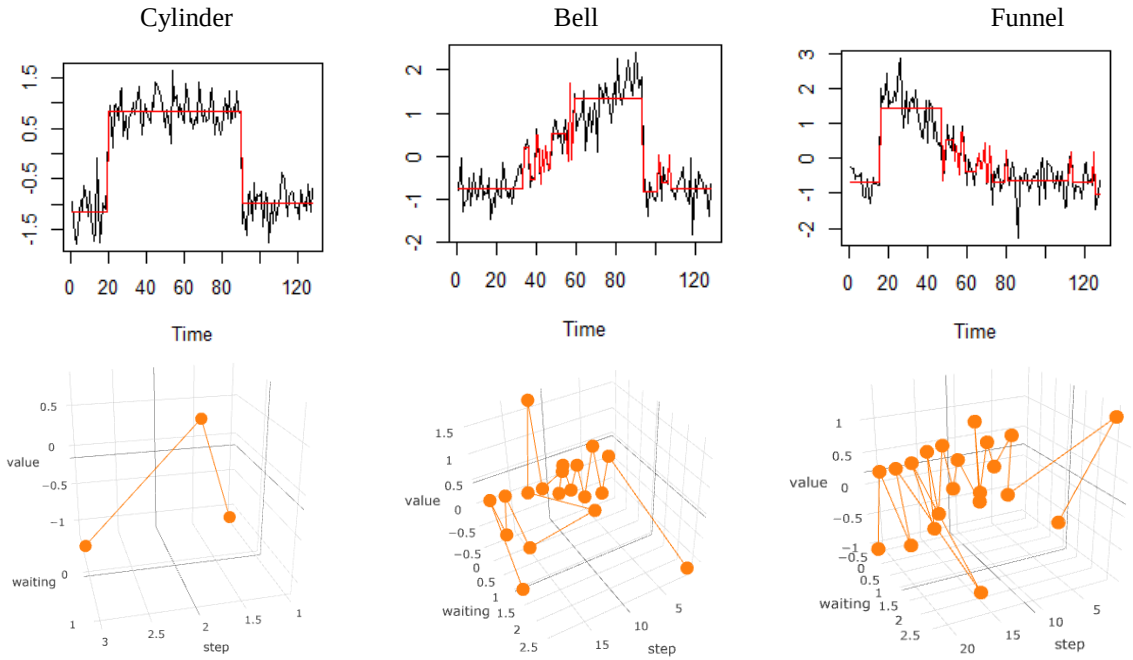


Figure. 1. Three classes of CBF time series dataset (Cylinder, Bell and Funnel): the raw data are shown in black, the segments are shown in red and CCA transformed time series in a 3-dimensional space are shown in orange.

Another contribution is introducing Blocked Dynamic Time Warping (BDTW), a new similarity measure which works on run-length encoded time series representation. BDTW utilizes repetitive values in time series to reduce DTW computation time and it can be used for the exact or approximation calculation of DTW. BDTW, has variations that provide an upper bound (BDTW UB) and a lower bound (BDTW LB) for DTW. BDTW UB is a close approximation of exact DTW, and it can be used as an approximation method. It is faster than traditional DTW for time series with high levels of values repetition. Moreover, BDTW can be combined with time series representation methods which provide constant segments to serve as an approximation method even for the time series without values repetition. We also show that in the exact DTW calculation,

for time series with high levels of repetitive values, BDTW UB and BDTW LB perform better than other popular upper and lower bounding techniques.

1.3 Outline of This Thesis

The rest of this research is organized as follows: In Chapter 2, background and related works are reviewed. Chapter 3, explains the time series representation and proposes a new representation technique called Control Chart Approximation. Chapter 4, explains 1-NN 3 Dimensional DTW, showcases two case studies, and compares the accuracy and running time of 1-NN 3D DTW against other state-of-the-art classifiers and its speed improvement techniques. Chapter 5 introduces Blocked DTW with its variants and applications. Finally, Chapter 6 concludes the thesis and talks about future work.

2 LITERATURE REVIEW

In this chapter, definitions, a review of time series classification models and review of Dynamic Time Warping are presented. Material from my own previously published work [73], 2017 IEEE, is reused in this chapter.

2.1 Definitions

Definition 1. A time series, $T\ (t_1, t_2, \dots t_n)$, is a sequence of *n* real values recorded over time.

Defining a similarity/distance measure between two time series is at the core of most time series data mining tasks.

Definition 2. For two time series X $(x_1, x_2, \dots x_n)$ and Y $(y_1, y_2, \dots y_n)$, Euclidian distance is defined as

$$ED(X,Y) = \sqrt[2]{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Euclidean distance is the most basic and classic distance measure which is still competitive in some problems and domains [1]. Figure 2, illustrates an example of the Euclidean distance visualization

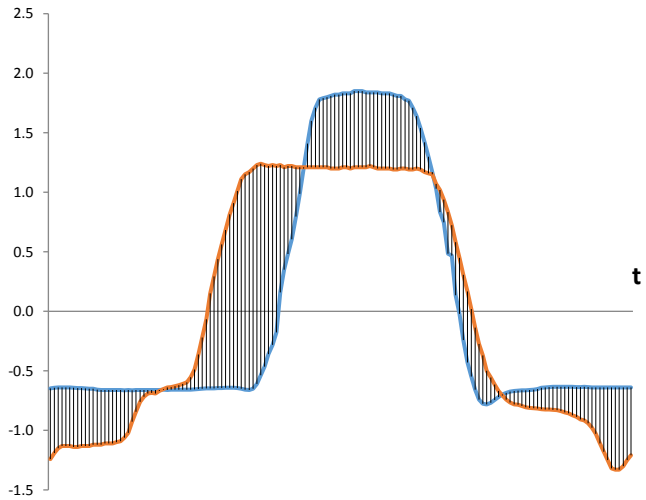


Figure 2.Visualization of Euclidean distance (linear alignment of the pair points) between two time series in blue and orange

Generally, raw time series contain various sorts of distortion and noise, which makes their comparison difficult. The common distortions and invariances are amplitude and offset invariance, local scaling ("warping") invariance, uniform scaling invariance, phase invariance, occlusion invariance and complexity-invariant distortions [10],[11].

Some invariances can be addressed and fixed easily. For example, normalizing raw data for the time series X can be used to prevent amplitude and offset invariance for this time series.

$$Z - norm(X) = (x_1', x_2', \dots, x_n')$$

where $x_i' = \frac{x_i - \mu}{\sigma}$, μ is the mean of X and σ is the standard deviation of X. In addition, a variety of distance function and data representation techniques have been developed to prevent different invariances. DTW is the most prominent example which addresses the local scaling invariance [2]-[5].

2.2 A Review of Time Series Classification Models

In [54] different time series classification methods and algorithm are categorized in 6 groups: Time Domain Distance Based Classifiers, Differential Distance Based Classifiers, Dictionary Based Classifiers, Shapelet Based Classifiers, Interval Based Classifiers and Ensemble classifiers.

Here we briefly review these categories and their classification models but our focus is on DTW because multiple rigorous independent studies [6,54] show that for the problem of time series classification, 1-NN DTW is very hard to beat.

Additionally, where NN-DTW can be beaten, it usually by a very insignificant margin, at the cost of enormous effort in coding or complexity of implementation, and a large time and space overhead.

2.2.1 Time Domain Distance Based Classifiers

Weighted DTW (WDTW) [16]: WDTW is a form of DTW that adds a weight to nearer neighbors. This weight depends on the difference of phases between a reference point and a query point. WDTW penalizes the points distances according to the phase difference.

*Time Warp Edit (TWE) [56]:*TWE is an elastic distance matric that contains features of the longest common subsequence and DTW. It contains a stiffness parameter that controls the elasticity. This parameter allows TWE to be a middle ground between the stiff Euclidian distance and the flexible DTW. Similar to WDTW, the stiffness imposes a multiplicative penalty on the distance between points. If sequences do not match, a penalty of λ is applied.

Move-Split-Merge (MSM) [57]: MSM distance offers a measure that is similar to common distance_based approaches, where dissimilarity/similarity is determined through Move and Split operations to transform a given time series. The MSM method offers increased robustness.

2.2.2 Differential Distance Based Classifiers

Complexity Invariant Distance (CID) [11]: CID calculates complexity differences between two time series and uses it as a correction factor in the distance measures. The complexity is measured using the sum of squares of the first difference of time series. As the difference in complexity of time series increase, the distance also increases.

Derivative DTW (DD_{DTW}) [58]: The DD-DTW uses a weighted combination of original time series distances and first order differences in a nearest neighbor classification. These two distances are combined with a weighted parameter. This parameter is determined by conducting a leave-one-out cross validation on the training set.

Derivative Transform Distance (DTD_c) [59]: DTDc is an extension of DD_{DTW} that uses DTW in conjuncture with derivatives.

2.2.3 Dictionary Based Classifiers

Bag of Patterns (BOP) [52]: BOP is a dictionary classifier that is constructed on the Symbolic Aggregate Approximation (SAX). This technique converts time series to sequence of letters. SAX reduces the length of a time series through Piecewise Aggregate Approximation (PAA). BOP works by applying SAX to a window of the time series to form a 'word'. If consecutive windows contain the same word, the second is deemed unnecessary and discarded. The distribution of words is used to form a count histogram. To classify new series, a 1-NN Euclidean distance classifier is used on the histograms.

Symbolic Aggregate Approximation – Vector Space Model (SAXVSM) [53]: SAXVSM creates word distributions over the classes and weights these by the term frequency / inverse

document frequency. 1-NN and word frequency distribution are used to predict the label of a new case.

Bag of SFA Symbols (BOSS) [47]: BOSS uses windows to form words over a time series but in a different way than BOP and SAXVSM. Instead of PAA, BOSS uses a truncated Discrete Fourier Transform (DFT). The truncated series is discretized using a method called Multiple Coefficient Binning (MCB). BOSS includes a parameter that determines whether the subseries are normalized or not.

DTW Features (DTW_F) [60]: DTWF combines DTW distances and SAX histograms using a feature generation scheme.

2.2.4 Shapelet Based Classifiers

Shapelets are time series subsequences that are discriminatory of the membership to a class. Shapelets can be used as the splitting criterion when applying a decision tree for classification.

Fast Shapelets (FS) [61]: FS is an extension of the decision tree shapelet that accelerates shapelet discovery. In FS a frequency count histogram is built using multiple random projections.

Shapelet Transform Ensemble (ST) [46]: ST splits the shapelet discovery by finding the top k shapelets on each run. ST can be utilized using kNN, Naïve Bayes, decision tree, support vector machine, etc. Each classifier is assigned a weight based on the cross validation training accuracy.

Learned Shapelets (LS) [55]: LS finds *k* shapelets using *k*-means clustering of candidates from the training data. A logistic loss function is the objective function according to a logistic

regression model for each class. The weights of the regression and shapelets are learned by the algorithm. A check is performed at certain intervals as to whether divergence has occurred.

2.2.5 Interval Based Classifiers

Interval Based Classifiers work based on extracting features from different intervals of each series. A problem that should be addressed in this approach is the huge dimension of the feature space.

Time Series Forest (TSF) [62]: TSF overcomes this problem by implementing a random forest method and considering summary statistics of each interval as features.

Time Series Bag of Features (TSBF) [49]: TSBF has multiple steps and is an extension of TSF.

Learned Pattern Similarity (LPS) [63]: LPS was developed by the same group as TSF and TSBF. The main difference is that subseries become attributes instead of cases.

2.2.6 Ensemble Classifier

Elastic Ensemble (EE) [64]: EE is a collaboration of nearest neighbor classifiers that use elastic distance measures. EE is built on the fact that none of the individual classifiers significantly outperform DTW. However, it can be seen that by combing the predictions of 1-NN classifiers to create a voting system, DTW can be outperformed.

Collective of Transformation Ensembles (COTE) [46]: COTE purposes a combination of classifiers from both the EE and ST domains. EE utilizes 35 classifiers and a voting system.

2.3 A Review of DTW

DTW allows nonlinear pairing of points in two sequences (Figure 3). As a result, DTW allows the two compared time series to be of different lengths - which is a feature our proposed model relies on as we explain in the next section.

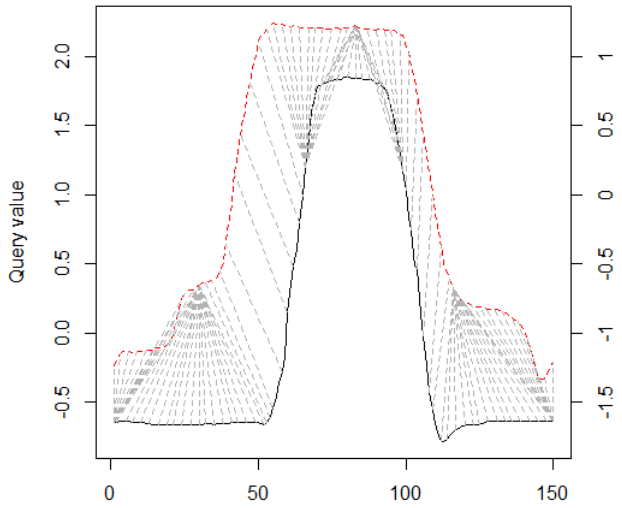


Figure 3. Nonlinear re-aligning of the pairs of points in DTW between two time series in red and black

DTW computes the minimum distance of two time series by dynamically matching the points in a non-linear fashion. To calculate the distance between the two time series of X $(x_1, x_2, \dots x_n)$ and $Y\,(y_1, y_2, \dots y_m)$ with corresponding length of N and M, DTW finds the minimum Minkowski distance with l_p norm over the allowed matching between points of the two time series. A *n*-by-*m* matrix can be constructed to cover all possible matching and alignments between points of x_i and y_j. Each element (i, j) of this matrix represents the distance between x_i and y_j when aligned together, defined by: $d(x_i, y_j) = \|(x_i - y_j)\|_p$

where $\|.\|_p$ represents the l_p norm. For example, for $p = 2$, $d(x_i, y_j) = |x_i - y_j|^2$.

W is defined as a continuous path of elements in the matrix (alignments) from the beginning (x_1, y_1) to the end (x_n, y_m) of time series. Thus, W is the mapping path between X and Y with the length K.

The l^{th} element of W is defined as $w_l = d(i,j)_l$; that is,

$$W = w_1, w_2, \dots, w_l, \dots, w_k$$

$$max(n,m) \leq K \leq n + m - 1$$

DTW finds permissible matching sequences and paths. Then it proceeds to find the optimal path with the minimum distance. Figure 4, shows a graphical example of warping matrix and the optimal warping path.

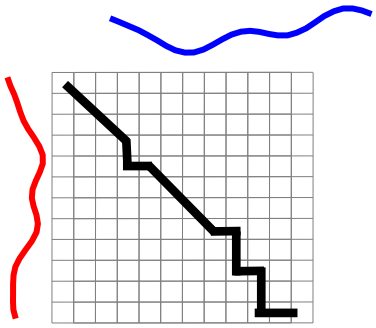


Figure 4. Warping matrix and the optimal warping path for two time series in red and blue.

Permissible *n*-by-*m* matching points, or alignment paths, are those that satisfy the following constrains [12]:

Boundary conditions: The path should not skip a part at the beginning or ending of the sequence. The first and the last matches should be (x_1, y_1) and (x_n, y_m) meaning $w_1 = (1,1)$ and $w_K = (N, M)$.

Continuity conditions: There should be no jumps in the path. The previous step for each point (i, j) in the path must be $(i-1, j)$, $(i, j-1)$ or $(i-1, j-1)$.

Monotonicity conditions: The warping path cannot go back in time. This means in each step of the warping path, i and j indexes stay the same or increase. Considering $w_k = (a, b)$, we have $w_{k-1} = (\acute{a}, \acute{b})$ where $a - \acute{a} \gg 0$ and $b - \acute{b} \gg 0$.

DTW must find the path which minimizes the warping distance:

$$DTW_p(X, Y) = min\left\{\sqrt[p]{\sum_{l=1}^{K} w_l}\right\}$$

The optimal path can be calculated using dynamic programming with the following recursive equation:

$$DTW_p(X, Y) = \sqrt[p]{\gamma(i, j)}$$

where γ(i, j) is calculated by: $\gamma(i, j) =$

$$\left|x_i - y_j\right|^p + min\{\gamma(i-1, j-1), \gamma(i, j-1), \gamma(i-1, j)\}$$

$$\gamma(0,0) = 0, \gamma(0, \infty) = \infty, \gamma(\infty, 0) = \infty$$

$$(i = 1,2, \ldots, n; j = 1,2, \ldots, m)$$

The value of the bottom right matrix cell ($\sqrt[p]{\gamma(N, M)}$) will contain the DTW distance of X and Y. Figure 5, demonstrates an example of direct and cumulative distances in the DTW cost matrix. The optimal path is shown in pink cells and (setting p = 2), the DTW distance is the square root of 5.

	5	3	2	2	4
5	0	4	9	9	1
6	1	9	16	16	4
5	0	4	9	9	1
4	1	1	4	4	0
3	4	0	1	1	1

(a)

	5	3	2	2	4
5	0	4	13	22	23
6	1	9	20	29	26
5	1	5	14	23	24
4	2	2	6	10	10
3	6	2	3	4	**5**

(b)

Figure 5. X={5,6,5,4,3},Y={5,3,2,2,4}. (a) shows the distance of aligned points of X and Y and (b) shows the cumulative distance matrix.

Several extensions of DTW have been proposed in the literature. Using the distance between derivatives instead of point to point distance was proposed in [13]-[15]. This method is called Derivative Dynamic Time Warping (DDTW). DDTW transforms the points of the time series to a higher level feature. For example, in [14] the transformation function for the point x_i in time series X is given as

$$X_A(d_i^a) = \frac{(x_i - x_{i-1}) + (\frac{x_{i+1} - x_{i-1}}{2})}{2}, \quad 1 < i < n$$

where *n* is the length of the time series *X*. Since the first and the last transformations are not defined, it is considered that $d_1^a = d_2^a$ and $d_n^a = d_{n-1}^a$.

The relative importance of the different phases of each alignment is considered in [16]. This method is called Weighted Dynamic Time Warping (WDTW). In WDTW, the distance between the two points x_i and y_j is defined as:

$$d_w(x_i, y_j) = \left\| w_{|i-j|}(x_i - y_j) \right\|_p$$

where w is a positive weight value between the two points x_i and y_j, and it is determined based on the phase difference$|i - j|$. In [16] a logistic weight function is proposed to calculate the $w_{|i-j|}$. For a phase difference of z, the weighting is

$$w(z) = \frac{w_{max}}{1 + e^{-g.(z-\frac{n}{2})}}$$

where $w_{max}$is the upper bound of the weight, n is the length of time series and g is the penalty for large phase differences. Then the optimum distance between the two sequences is defined as:

$$WDTW_p(X,Y) = \sqrt[p]{\gamma(i,j)}$$

where $\gamma(i,j)$ is calculated by

$$\gamma(i,j) = \left|w_{|i-j|}(x_i - y_j)\right|^p + min\{\gamma(i-1,j-1), \gamma(i,j-1), \gamma(i-1,j)\}$$

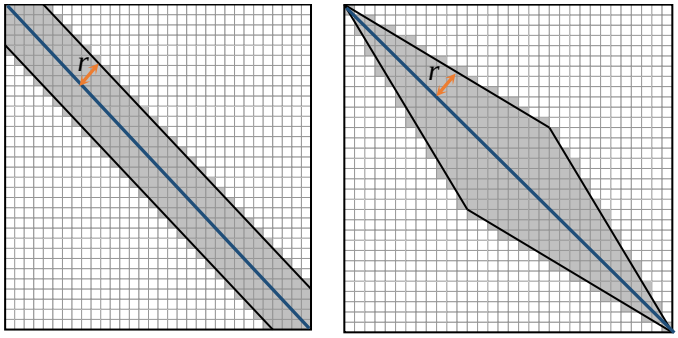
In [17], the weighted DTW concept is combined with DDTW, and it is called Weighted Derivative Dynamic Time Warping (WDDTW). Different variations of DTW often outperform each other in different application domains. It has been shown that 1-NN DTW is one of the best distance-based classification techniques and exceptionally hard to beat [6].

2.4 Improving the Speed of DTW

2.4.1 Global Warping Constraints

The idea of warping constraints is to limit the wandering distance of the search paths by defining a band (warping scope) in the DTW warping matrix. This band does not necessarily retrieve the optimal path, but it is expected to provide a decent path. It is expected that a decent path is close to the matrix diagonal, and warping constraints are defined around the diagonal.

The Sakoe-Chiba band [2] and the Itakura Parallelogram [18] are the most common constraint methods. Global constraints speed up the calculations, however, the classification time and accuracy depends on a parameter (*r*), where *r* is the allowed range of warping from the diagonal of the matrix. Figure 6, shows a graphical example of constraints for the Sakoe-Chiba band and the Itakura Parallelogram. Cross validation on a training dataset is used to learn the best constraint (*r*). The classifier which uses 1-NN DTW with a learned warping constraint on cross validation increases accuracy in some problems [19], but training process is time expensive.



Sakoe-Chiba band Itakura Parallelogram

Figure 6. Example of global constraints

2.4.2 Lower Bounding

The idea of lower bounding is to do the expensive full calculation of optimal path in DTW matrix, only when it is absolutely unavoidable. Applying lower bound speeds up the nearest neighbor search by pruning off unhopeful candidates [6],[21]. LB- Kim, LB- Yi and LB- Keogh are the most common lower bound for time series classification which are graphically presented on an example for candidate C and query Q time series in Figure 7. LB- Kim [22] is the total summation of the squared differences between the two sequences' first (A), last (D), minimum (B) and maximum points (C). These distances are presented in Figure 7a. LB- Yi [24] is calculated as:

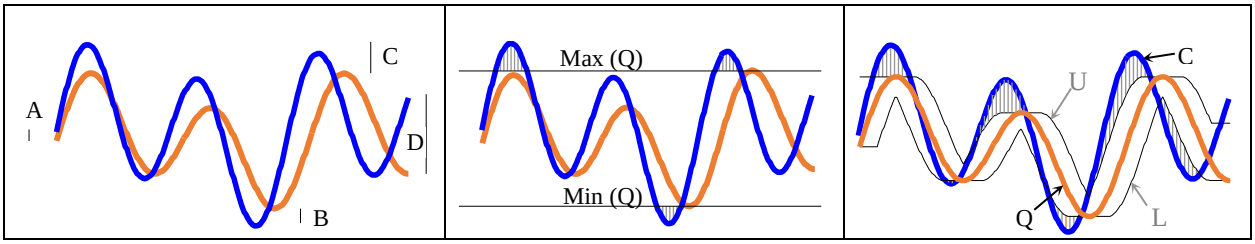
$$LB-Yi(Q,C)=\sqrt{\sum_{i=1}^{n}\begin{cases}(Min_{j=1}^{m}(Q_j)-C_i)^2 & if\ C_i<Min_{j=1}^{m}(Q_j)\\ (C_i-Max_{j=1}^{m}(Q_j))^2 & if\ C_i>Max_{j=1}^{m}(Q_j)\\ 0 & otherwise\end{cases}}$$

The sum of the squared length of gray lines in Figure 7b, represents LB- Yi.

LB- Keogh bound is explained in details in [6],[22],[24]. LB- Keogh first defines U and L as the upper and the lower bound of query time series as: $U_i = \max(q_{i-r}: q_{i+r})$ and $L_i = \min(q_{i-r}: q_{i+r})$ where r is the window size. Then LB- Keogh is formulated as:

$$LB-Keogh(Q,C)=\sqrt{\sum_{i=1}^{n}\begin{cases}(c_i-U_i)^2 & if\ c_i>U_i\\ (c_i-L_i)^2 & if\ c_i<L_i\\ 0\end{cases}}$$

The sum of the squared length of gray lines in Figure 7c, represents LB- Keogh.



a) LB- Kim b) LB- Yi c) LB- Keogh

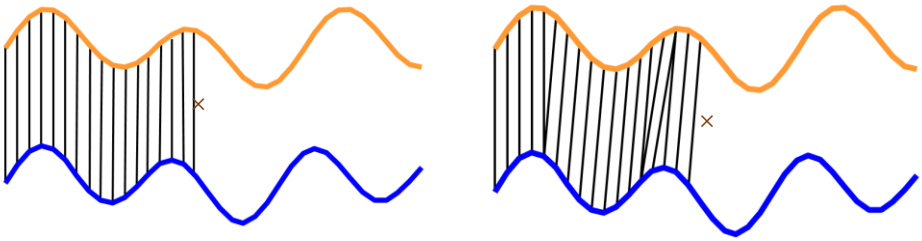
Figure 7. Example of lower bounding techniques.

A review and evaluation of lower bounding methods used for DTW is presented in [42].

2.4.3 Early Abandoning

Early abandoning is another method to accelerate 1-NN DTW calculations (as well as the other distance based classifiers such as 1-NN ED). This is done by calculating partial distance

accumulation for a candidate sequence, and comparing it to a threshold which is the best-so-far candidate. If partial accumulation goes beyond the threshold (best-so-far) at any time, the calculation is terminated, and the candidate is discarded. This technique reduces redundant calculations in the similarity search. Early abandoning was proposed and applied in [4] and [21] to accelerate the calculation of ED, as well as in [5] and [26], to speed up 1-NN DTW calculations.



(a) (b)

Figure 8. Early abandoning technique. (a) Early abandoning on ED and (b) Early abandoning on DTW.

×: stop the calculations, partial distance accumulation exceeds best-so-far

3 TIME SERIES REPRESENTATION WITH CONTROL CHART APPROXIMATION

In this chapter, Control Chart Approximation method is introduced. Material from my own previously published work [73], 2017 IEEE, is reused in this chapter.

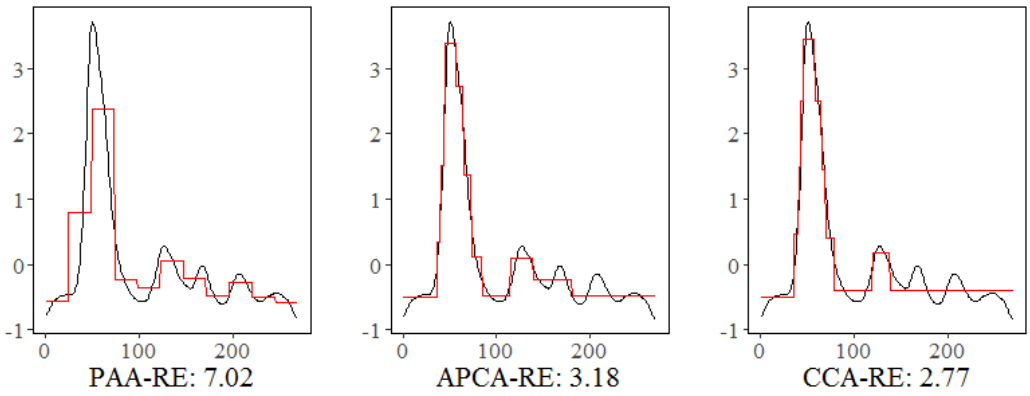
3.1 Introduction

Each value in a time series can be considered as a dimension. Time series are typically high dimensional data (i.e. long length). Time series representations project points into a new space where it can be processed more efficiently. Time series representation techniques (also called dimensionality reduction techniques) aim to reduce the dimensionality (length) of time series by extracting features from the raw data. Time series representation should follow the following goals [21]: dimensionality reduction, storage reduction, noise removal, computational costs reduction, and minimizing information loss (the reconstruction error). A review of representation methods, as well as distance based classification techniques, is presented in [6]. Time series representation in the literature can be mainly divided into two groups of non-symbolic and symbolic representations. Some of the non-symbolic representation methods are listed in [6], including Discrete Fourier Transformation [27], Single Value Decomposition [28], Discrete Wavelet Transformation [29], Piecewise Aggregate Approximation (PAA) [30], Adaptive Piecewise Constant Approximation (APCA) [31], and Chebyshev polynomials [32]. Reference [33] lists some of the symbolic representation methods, including Shape Description Alphabet [34], Interactive Matching of Patterns with Advanced Constraints in Time Series Databases [35],

Clipping [36], Persist [37], Piecewise Vector Quantized Approximation [38], and Symbolic Aggregate Approximation (SAX) [39] and its extensions [40],[41].

This work proposes a new numeric representation of time series denoted as the Control Chart Approximation (CCA). CCA is discussed in detail in Section 3.3. An example is presented in Figure 9, to compare CCA with PAA and APCA. Euclidean distance between the raw time series and the approximation is the reconstruction error (RE). In this example, it can be seen that after approximation with the same compression ratio (14 segments), CCA has the lowest RE.

The output of CCA looks graphically similar to the output of PAA and APCA. A time series is divided into flat segments. In PAA, the length of the segments must be same, but APCA and CCA relax this constraint. The value of each segment is the mean of all the points which fall within that segment.



(a) RR Lyrae Variables Class (b) Eclipsed Binaries Class (c) Cepheids Class

Figure 9. Comparison of approximation methods and their reconstruction error. The raw data is in black and approximation is in red.

3.2 Related Work

There have been multiple contributions which focus on the reduction of data dimensionality by obtaining constant segments approximation. Here, we review two popular methods which give the constant segments: Piecewise Aggregate Approximation (PAA), and Adaptive Piecewise Constant Approximation (APCA).

3.2.1 Piecewise Aggregate Approximation

Let n be the length of a time series $X = x_1, x_2, \ldots, x_n$, and w be the dimensionality of the transformed time series ($1 \leq w \leq n$). Essentially, to reduce the length of a time series from n to w, begin by dividing the time series into w equal length segments. The average value of the points that lie within each segment is calculated and stored into a vector.

A time series X of length n can be represented with a w-dimensional vector $\overline{X} = \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_w$, where The i^{th} element of this vector $\bar{x}$ is calculated by:

$$\bar{x}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{(\frac{n}{w})i} x_j$$

Note that when $w = n$, the transformed representation remains like the original time series, and when $w = 1$, the transformed sequence is the mean of the original time series.

In PAA, i^{th} segment can be considered as $\bar{x}_i$ value being repeated $\frac{n}{w}$ times on that segment.

3.2.2 Adaptive Piecewise Constant Approximation

Though PAA is a competitive time series transformation technique that has been proven with even the most complex transforms for indexing. Adaptive Piecewise Constant

Approximation (APCA) attempts to improve the performance quality of the PAA approximation by permitting the segments to have arbitrary lengths as opposed to fixed [5][6]. To achieve this, APCA stores two numbers: the value of the mean of all points in each segment as well as the length of the segment. For a time series $T = \{t_1, t_2, \ldots, t_n\}$, we can reduce its length via APCA, resulting in:

$$T = \text{v}\ \{<tv_1, tr_1>, <tv_2, tr_2>, \ldots, <tv_w, tr_w>\} \quad tr_0 = 0$$

where tv_i is the segmented average of the i^{th} region, and tr_i is the right most data point of the i^{th} region. Instead of applying the length of the segment, the location of its endpoint is stored. The length of each segment can still be calculated by the difference of each endpoint:

$$\text{Length of segment } i = tr_i - tr_{i-1}$$

In APCA, i^{th} segment can be considered as tv_i value, being repeated $(tri - tri - 1)$ times on that segment.

3.2.3 Control Charts

CCA uses control thresholds to detect the shifts in time series values. The idea is originated from control chart monitoring which used in statistical quality control. The control charts are frequently employed to monitor and regulate quality characteristics of a process that has been measured or recorded over time [17], and hence, are very similar to time series. The control chart's center-line represents the mean value of the quality characteristic and the two horizontal lines above and below are known as the upper control limit (UCL) and the lower control limit (LCL), respectively. When recorded measurements are within the two control limits, the process is said to be in control. If a point ventures beyond the threshold of the control limits, it indicates that the

process mean has shifted, and may be out of control. Figure 10, displays an example of a control chart.

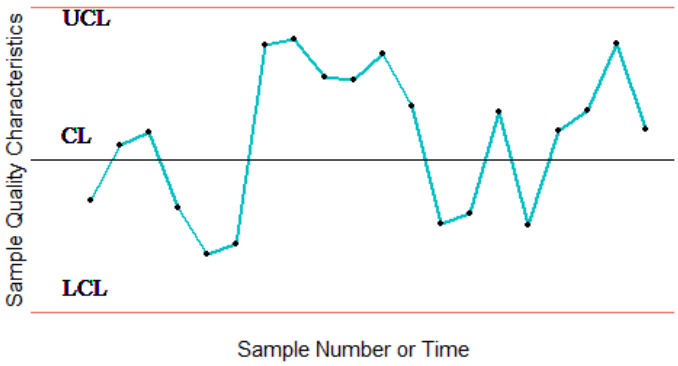


Figure 10. Control chart

Control limits are usually defined based on a fixed distance from the center-line. If we consider μ as mean of a specific quality characteristic, and σ as the standard deviation of it, then the center-line and control limits can be defined as following:

$$Center\ line = \mu$$
$$UCL = \mu + L\sigma$$
$$LCL = \mu - L\sigma$$

where L represents the distance (in the terms of standard deviation) between the control limits and the center line. These kinds of control charts are often referred to as a Shewhart control chart. Typically, three-sigma limits are employed (i.e., L is usually set to 3).

When data points are z-normalized - as it is the common practice for the time series data in order to prevent amplitude and offset invariance [11] - the center-line will be 0, and UCL and LCL will be 3 and -3, respectively. An example of a standardized control chart is presented in Figure 11.

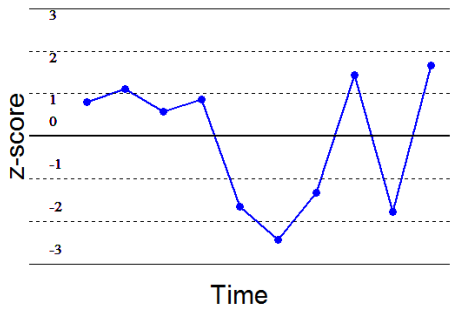


Figure 11. Standardized control chart

3.3 Control Chart Approximation: A time series representation

In this section, we present *Control Chart Approximation (CCA)*, our proposed numeric representation of time series. Given a time series, $X = (x_1, x_2, \dots x_n)$, a CCA representation of X is made as follows:

$$C_X = \{(t_1, v_1, u_1), \dots, (t_i, v_i, u_i), \dots, (t_k, v_{k,}, u_k)\} \quad (3.1)$$

$$i = 1,2,..,K$$

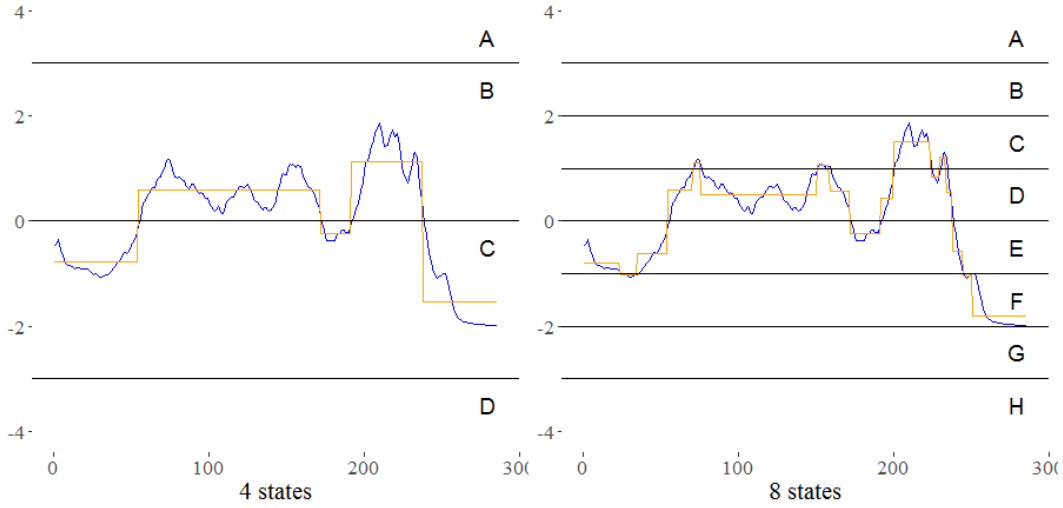
where the tuple (t_i, v_i, u_i) defines the *ith* segment, t_i is the start time, v_i is the value, u_i is the duration of *the* segment in the time series, and *K* is the total count of segments.

Each segment contains a continuous sequence of data points in the time series whose values were falling into the same range (state). The segments embody the continuous state change of the time series values. Similar to control limits in control charts, certain thresholds need to be defined in order to detect such state changes over time. A segment starts when a data point passes a threshold and its time value marks the beginning of a new segment (t_i). We call these special points - at the beginning and end of the segments - the *jump points*. The value v_i of each segment is defined as the mean of the raw data points falling in that segment. The value and duration of each segment are defined as follows:

$$u_i = t_{i+1} - t_i \qquad (3.2)$$

$$v_i = \left(\sum_{j=t_i}^{t_{i+1}} x_j \right) / u_i \qquad (3.3)$$

Figure 12a, shows a standard control chart with ±3 /sigma control limits used to partition a time series into 4 states (A, B, C and D) and results in a CCA representation of the time series with 5 segments. In this example, the standard control limits of ±3 have been used, and the range between the two control limits have been divided to 2. For more exact approximations, the same concept can be used by dividing the space between two control limits into more partitions.



a) b)

Figure 12. Time series approximation. The raw time series is presented in blue and its approximation in orange.

This number of partitions between control limits in CCA is controlled by the parameter *s*, which is the distance between the thresholds within the upper and lower control limits.

In Figure 12b, the distance between controls limits, *s* is set to 1, which creates 8 states. This results in a more accurate approximation, however, it increases the number of produced segments by 12 (17 segments). Algorithm 3.1, represents how CCA algorithm detects the jump points and produce the segments in detail.

The next step in CCA is normalizing the segments durations to have the same the scale as segments values. Thereby, each segment duration (u_i) is z-normalized by relative to mean (μ_u) and standard deviation (σ_u) of all the segments durations. We denote the normalized duration of each segment as its ***waiting time***(w_i).

$$w_i = {(u_i - \mu_u)}/{\sigma_u} \tag{3.4}$$

Algorithm 3.1 The CCA algorithm with z-normalized Time Series X, control limit K and distance parameter s

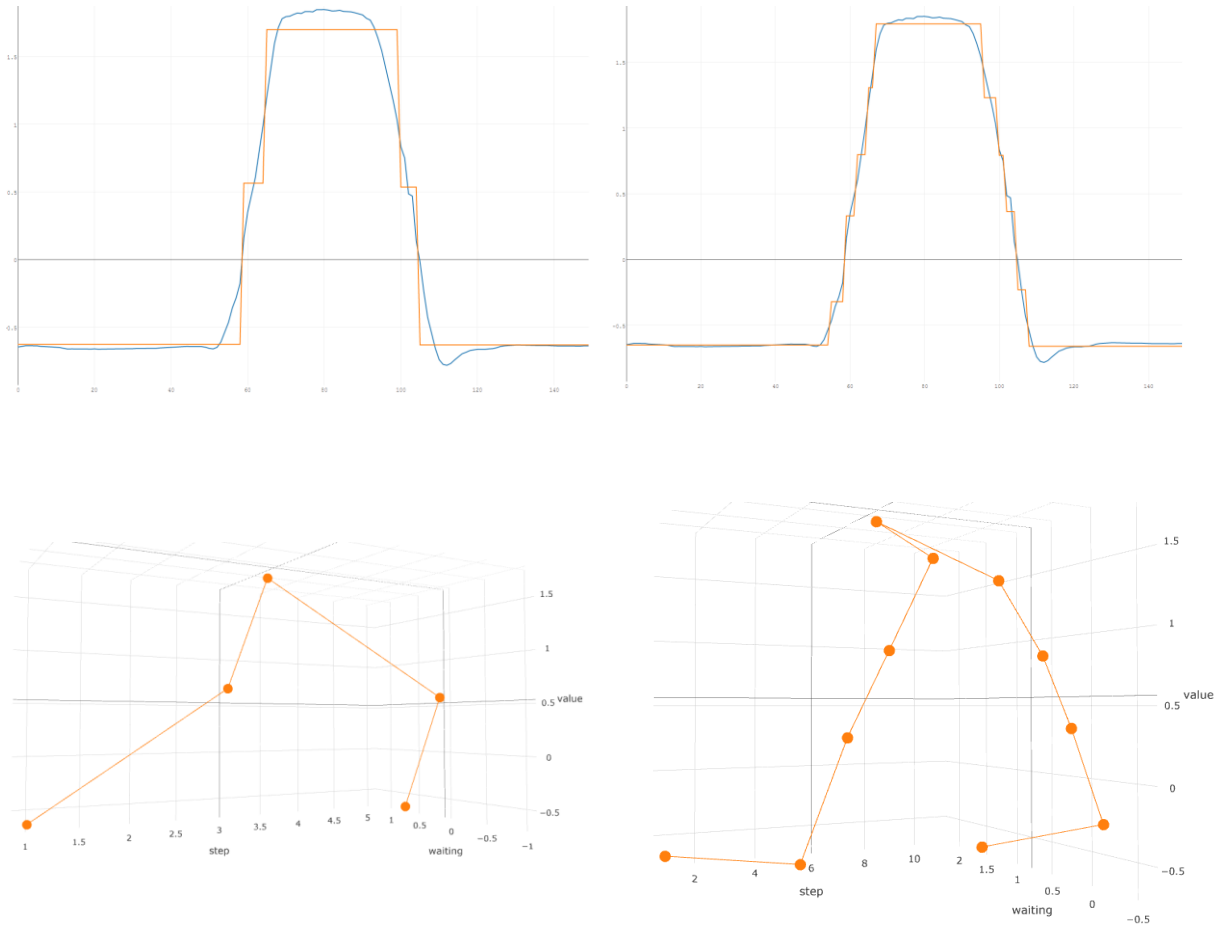
1. **function** Compute_CCA(X=$\{x_1, \dots, x_n\}, K, s$)
2. j ← 1, t_j← 0
3. $C_X \leftarrow \emptyset$
4. **if** $n = 1$ **then return** C_X=$\{(0, x_1, 1)\}$
5. **end if**
6. for $i \leftarrow 1 \dots n$ do
7. Find $r_i \in \{(-\infty, -K], (-K, -K + s], \dots, (\dots, K], (K, \infty)\}$ such that $x_i \in r_i$
8. **if** $(i > 1$ & $r_i \neq r_{i-1})$ **or** $i = n$ **then**
9. $t_{j+1} \leftarrow i$ # start of the (j+1)th segment
10. $u_j = t_{j+1} - t_j$
11. $v_j = (\sum_{i=t_j}^{t_{j+1}} x_i)/\ u_j$
12. $C_X \leftarrow C_X \cup (t_j, v_j, u_j)$
13. **end if**
14. **end for**
15. **return** C_X
16. **end function**

The final result of CCA are sequential normalized segment values (*v*) and waiting times (*w*).

$$C_X = \{(t_1, v_1, w_1), \dots, (t_i, v_i, w_i), \dots, (t_k, v_{k,}, w_k)\} \tag{3.5}$$

The graphical presentation of this output can be shown in 3-dimensional graphs. In Figure 13, CCA time series representation is presented with two different *s* values. In the upper left graph, the threshold distance parameter is set to *1* (*s=1*); it has created 5 segments. Then, the segments durations are normalized. The lower left graph shows the 3-dimensional representation of the final

CCA segments. In the upper right graph, the CCA representation of the same time series with *s = 0.5* shows 11 resulting segments. The lower right graph shows the final values after normalization in 3D space.



a) *s*=1 b) *s*=0.5

Figure 13. CCA examples (a)s=1 and (b) s=0.5.

In CCA the value of *s* can vary from 3 to ε. The lower the *s*, the more states and, expectedly, more segments it will create over time. Table I, shows the corresponding *s* values for some desired number of states.

TABLE I. A SAMPLE OF S POSSIBLE VALUES AND THEIR CORRESPONDING NUMBER OF STATES

Number of states	*s*	*Number of states*	*s*
4	3	11	0.67
5	2	12	0.60
6	1.5	13	0.55
7	1.2	14	0.50
8	1	15	0.46
9	0.86	16	0.43
10	0.75	…	…

When *s* is extremely small (ε), the smallest change in the value of each point compared to its predecessor will make that point a jump point. In this special case - with the exception of constant remaining points – every point in the time series will become a jump point, and the resulting CCA will be very close to, or the same as, the original time series. In the process of time series classification, the best choice of *s* is dependent on the structure of data. In order to find optimal *s* for a specific time series data, cross validation can be used to learn the best choice of *s* for that problem.

4 THREE DIMENSIONAL DTW DISTANCE

In this chapter, 3 Dimensional Dynamic Time Warping (3D DTW), Lower Bounding and Early Abandoning on 3D DTW are presented. Material from my own previously published work [73], 2017 IEEE, is reused in this chapter.

4.1 3D DTW Calculation

Suppose we have two time series, C and Q, and we want to measure the distance of these time series after CCA representation (transformation). In order to calculate this distance, we propose a new version of DTW called 3 dimensional DTW (3D DTW) which is an adaptation of traditional DTW on more axis. Suppose the transformed time series are

$$Transformed\ C = \{(v_1^c, w_1^c), \dots, (v_i^c, w_i^c) \dots (v_n^c, w_n^c)\}$$

$$Transformed\ Q = \{(v_1^q, w_1^q), \dots, (v_j^q, w_j^q) \dots (v_m^q, w_m^q)\}$$

Note that we eliminated the t_i values in the tuples, as they are not considered as a dimension for DTW calculation. Similar to conventional DTW, the segments are ordered along the matrix based on their occurrence in time, and time distance is inherent in DTW path calculation. Hence, we do not consider the starting time of each segment in the distance calculation.

Note that even if two time series are from the same problem, and if the lengths of the original time series are equal, after CCA transformation they will not necessarily have equal lengths (the same number of segments/steps).

First, we define a distance matrix between every two segment. Each element of the matrix is defined as:

$$D(i,j) = (v_i^c - v_j^q)^2 + (w_i^c - w_j^q)^2, \qquad \text{where } i \in [1,n] \text{ and } j \in [1,m]$$

Then, a matrix is constructed based on pairwise distance $D(i,j)$ of segments starting from $\gamma(1,1)$ to $\gamma(N,M)$ similar to conventional DTW.

$\gamma(i,j)$ is calculated by:

$$\gamma(i,j) = (v_i^c - v_j^q)^2 + (w_i^c - w_j^q)^2 + min\begin{Bmatrix}\gamma(i-1,j-1)\\ \gamma(i,j-1)\\ \gamma(i-1,j)\end{Bmatrix}$$

$$\gamma(0,0) = 0, \gamma(0,\infty) = \infty, \gamma(\infty,0) = \infty \qquad (4.1)$$

$$(i = 1,2,\dots,n; j = 1,2,\dots,m)$$

After calculating all elements of the matrix, the square root of the last element, $\gamma(N,M)$ is the distance between transformed C and Q.

3D DTW follows the same conditions (i.e. Boundary conditions, Continuity and Monotonicity) and the same algorithm as regular DTW, but the distance matrix is changed according to the structure of the transformed time series. We demonstrate an example here in order to illustrate the technique clear.

Example 1: We randomly selected two time series from the *Gun_Point* train dataset from [43]. The original time series are shown in Figure 14. We are interested in calculating their 3D DTW distance after applying the CCA transformation. Setting *s* equal to 1, CCA results in the following segment list. Figure 15, shows these time series after the transformation in a 3-dimensional space.

$$Transformed\ Q$$

$$= \{(-0.56,1.28), (0.59,-0.67), (1.51,-0.76), (2.03,-0.09), (1.52,-0.67), (0.5,-0.81), (-0$$

$$Transformed\ C$$

$$= \{(-0.66,1.38), (0.48,-0.76), (1.66,0.71), (0.47,-0.67), (-0.63,1.43)\}$$

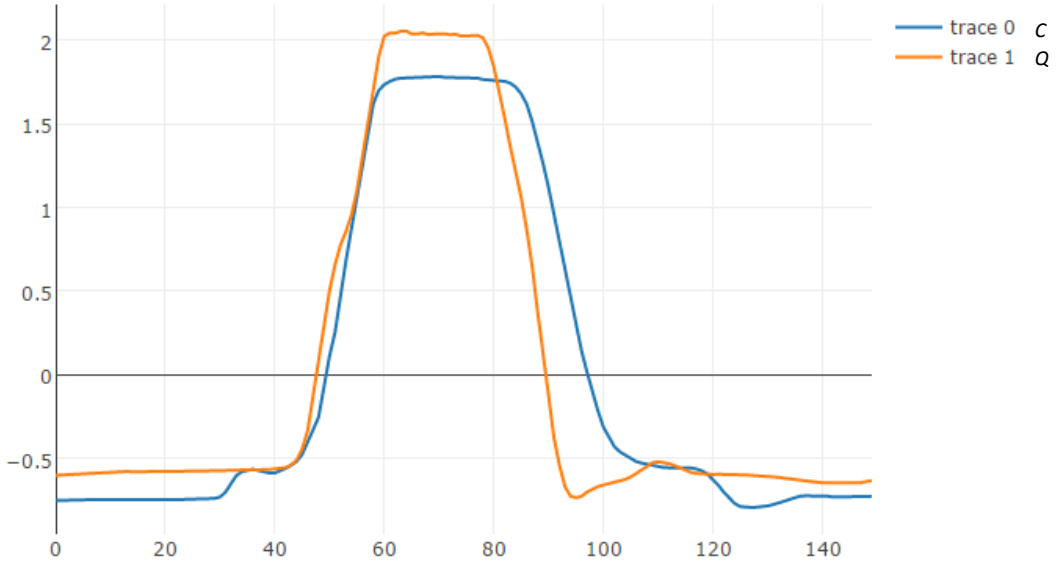


Figure 14. Original time series of C and Q.

Table II, shows the matrix for calculating the distance of two time series after CCA transformation, and the alignment between points.

The first element $\gamma(1,1)$ in this matrix is calculated as:

$\gamma(1,1) = (v_1^c - v_1^q)^2 + (w_1^c - w_1^q)^2 = ((-0.66) - (-0.56))^2 + (1.38 - 1.28)^2$ which is 0.02.

Then the first row is calculated as:

$$\gamma(1,j) = (v_1^c - v_j^q)^2 + (w_1^c - w_j^q)^2 + \gamma(1,j-1) \qquad (4.2)$$
$$j = 1,2,..,m$$

For example,

$\gamma(1,2) = (v_1^c - v_2^q)^2 + (w_1^c - w_2^q)^2 + \gamma(1,1) = ((-0.66) - 0.59)^2 + (1.38 - (-0.67))^2 + 0.02$, which is 5.750.

Then, the first column is calculated as:

$$\gamma(i,1) = (v_i^c - v_1^q)^2 + (w_i^c - w_1^q)^2 + \gamma(i-1,1) \qquad (4.3)$$
$$i = 1,2,..,n$$

For example,

$\gamma(2,1) = (v_2^c - v_1^q)^2 + (w_2^c - w_1^q)^2 + \gamma(1,1) = (0.48 - (-0.56))^2 + ((-0.76) - 1.28)^2 + 0.02$, which is 5.280.

The remaining matrix elements, from ([2], [2]) to ([5], [7]), are calculated using formula 4.1.

For example,

$\gamma(2,2) = (v_2^c - v_2^q)^2 + (w_2^c - w_2^q)^2 + min\{\gamma(1,1), \gamma(2,1), \gamma(1,2)\} = (0.48 - 0.59)^2 + ((-0.76) - (-0.67))^2 + min\{0.02, 5.28, 5.75\}$, which is 0.040.

The value of the last element of the matrix i.e. $\gamma(5,7)$, shows the squared distance between these transformed time series. The arrows in the matrix show the optimal selected path and the aligned points. The aligned points are connected together with gray dash lines in Figure 15.

TABLE II. 3D DTW COST MATRIX AFTER CCA TRANSFORMATION OF TIME SERIES

	Q	(-0.56,1.28)	(0.59,-0.67)	(1.51,-0.76)	(2.03,-0.09)	(1.52,-0.67)	(0.5,-0.81)	(-0.6,1.81)
C	step	[1]	[2]	[3]	[4]	[5]	[6]	[7]
(-0.66,1.38)	[1]	0.020	5.750	15.000	24.410	33.320	39.440	39.620
(0.48,-0.76)	[2]	5.280	0.040	1.090	3.950	5.040	5.040	12.810
(1.66,0.71)	[3]	10.530	3.090	2.240	1.880	3.800	7.480	11.370
(0.47,-0.67)	[4]	15.380	3.110	3.330	4.670	3.000	3.020	10.270
(-0.63,1.43)	[5]	15.400	8.970	12.450	12.740	12.000	9.260	3.160

One of the advantages of 3D DTW is that all of the speed improvement techniques (such as the squared distance, warping constrain, lower bounding, early abandoning, etc.) in regular DTW, as well as DTW extensions techniques (such as DDTW, WDTW, WDDTW, etc.), are applicable on 3D DTW after minor changes in the algorithm. In the following section, we show how to apply lower bounding and early abandoning in 3D DTW.

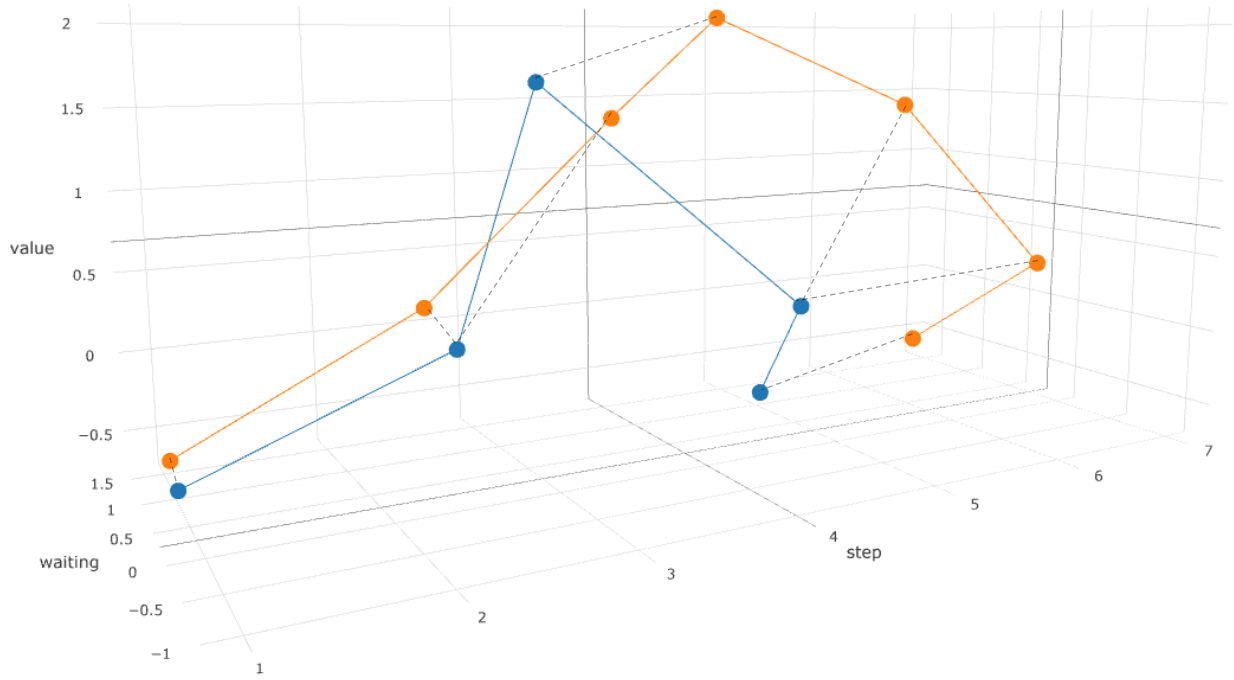


Figure 15. CCA transformed time series of C (in blue) and Q (in orange) and their 3D DTW alignment

4.2 Lower Bounding on 3D DTW

In order to demonstrate the lower bounding in 3D DTW, we start by applying LB -KIM for 3D DTW. In traditional DTW, LB -KIM for two time series is the total summation of the squared differences between the two sequence's first points(A), last points(D), minimum points (B) and maximum points (C).

In 3D DTW, LB -KIM is calculated based on the differences of the same points. The difference between the first and the last points can be calculated as:

$$Difference\ between\ the\ first\ points:$$

$$A = \sqrt[2]{(v_1^c - v_1^q)^2 + (w_1^c - w_1^q)^2} \tag{4.4}$$

$$Difference\ between\ the\ last\ points:$$

$$D = \sqrt[2]{(v_n^c - v_m^q)^2 + (w_n^c - w_m^q)^2} \tag{4.5}$$

In Figure 16, the difference between the first points (A) and the last points (D) of the two sample transformed time series are shown.

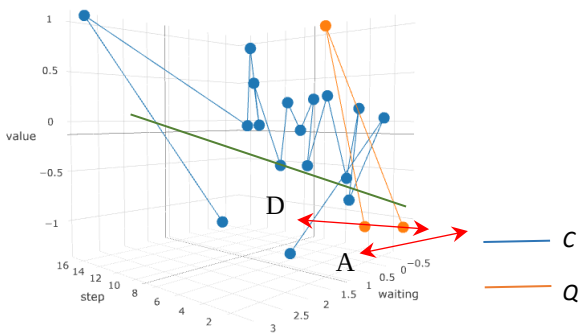


Figure 16. Difference between the first points (A) and the last points (D) of two sample transformed time series C and Q

Before calculating the distance between the maximum and minimum points, we identify them. The maximum points and the minimum points should be calculated based on a reference. The reference is a line in which the waiting time and the value of all the points are zero. This reference line of the example is presented in Figure 16, as a green line. The distance of each point from the reference can be calculated as:

$$D(T)_i = \sqrt[2]{{t_{v_i}}^2 + {t_{w_i}}^2} \qquad (4.6)$$

where $D(T)_i$ is the distance to the reference for the *ith* point of the transformed time series, *T*, and t_{v_i} and t_{w_i} are the values and waiting times of that point.

In order to make it more visible in Figure 17, we rotated the Figure 16, graph in a way that the reference line looks like a point. The distance of each point to this point (where waiting time and value are zero) is the distance to the reference. For example, the distance between the second point of time series *Q* and the reference is shown in Figure 17.

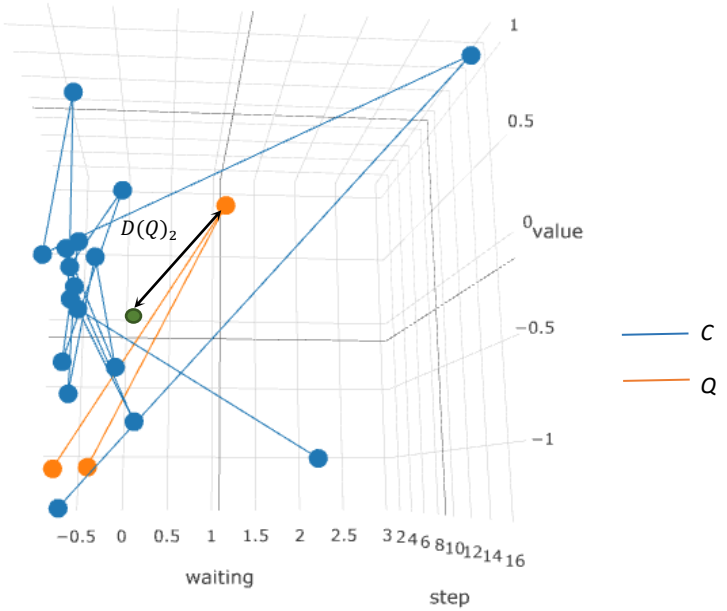


Figure 17. Distance of the second point of time series Q to the reference line.

Now, the distance of each point of a transformed time series to the reference can be calculated, and then the minimum and the maximum points can be identified based on these distances. Note, after detecting these minimum/maximum points, the lower bound of the difference between two minimum points (or two maximum points), is the difference between their distances to the reference. We need to prove that the difference between the distances, of each two points, to the reference is the lower bound of their direct distance. Here the difference between the distances, of each two points, to the reference is $|d_1 - d_2|$ and the direct distance is $\sqrt{(v_1 - v_2)^2 + (w_1 - w_2)^2}$ and we need to prove:

$$|d_1 - d_2| \leq \sqrt{(v_1 - v_2)^2 + (w_1 - w_2)^2}$$

where d_1 and d_2 are respectively the distance from point 1 (with value of v_1 and waiting time of w_1) and the distance from point 2 (with value of v_2 and waiting time of w_2) to the reference.

$$(d_{1-} - d_2)^2 \leq (v_1 - v_2)^2 + (w_1 - w_2)^2$$

$$\left(\sqrt{{v_1}^2 + {w_1}^2} - \sqrt{{v_2}^2 + {w_2}^2}\right)^2 \leq (v_1 - v_2)^2 + (w_1 - w_2)^2$$

$${v_1}^2{v_2}^2 + {v_1}^2{w_2}^2 + {w_1}^2{v_2}^2 + {w_1}^2{w_2}^2 - {v_1}^2{v_2}^2 - {w_1}^2{w_2}^2 - 2{v_1}^2{v_2}^2{w_1}^2{w_2}^2 \geq 0$$

$$(v_1 w_2 - w_1 v_2)^2 \geq 0$$

Therefore, we can calculate the lower bound of the distance between minimum points (as well as the lower bound of the distance between maximum points) by finding the difference between their distances to the reference:

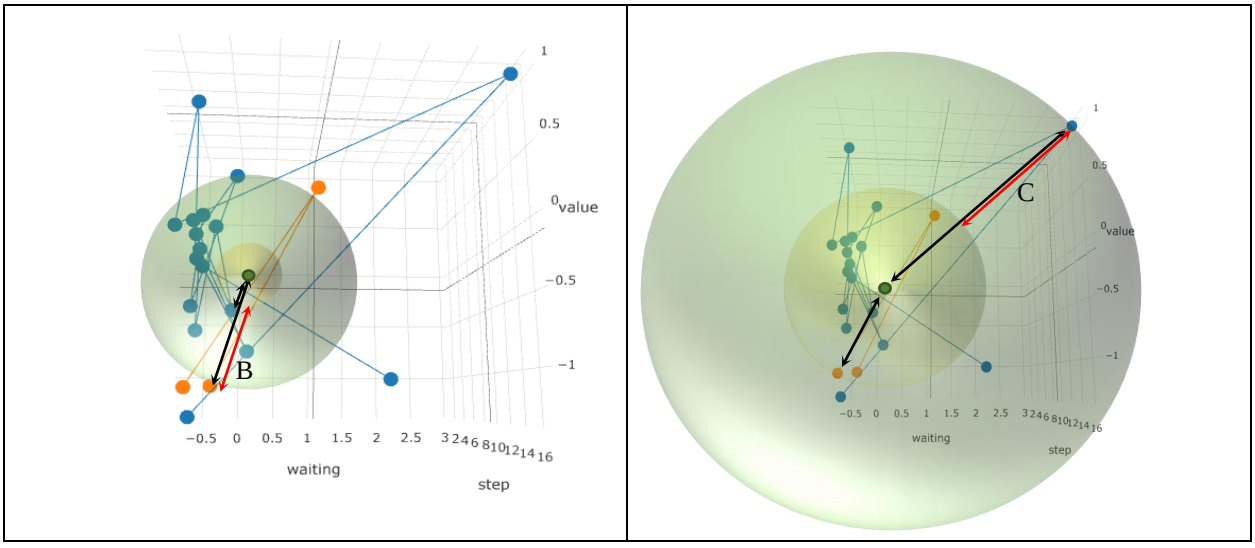
$$Difference\ between\ the\ Minimum\ points:$$

$$B = Min_{i=1}^{n}(D(C)_i) - Min_{j=1}^{m}\left(D(Q)_j\right) \tag{4.7}$$

$$Difference\ between\ the\ Maximum\ points:$$

$$C = Max_{i=1}^{n}(D(C)_i) - Max_{j=1}^{m}\left(D(Q)_j\right) \tag{4.8}$$

Figure 18a and Figure. 18b show the distance between minimum points (*B*) and the distance between maximum points (*C*), respectively.



(a) (b)

Figure 18. (a) The distance between minimum points (B) and (b) the distance between maximum points (C).

Lastly, the Kim- LB in 3D DTW, will be the summation of the squared values of these distances (*A, B, C* and *D*) from formula 4.4,4.5,4.6 and 4.7. Where A,B,C and D are the difference between the start points, the minimum points, maximum points and end points of the time series respectively.

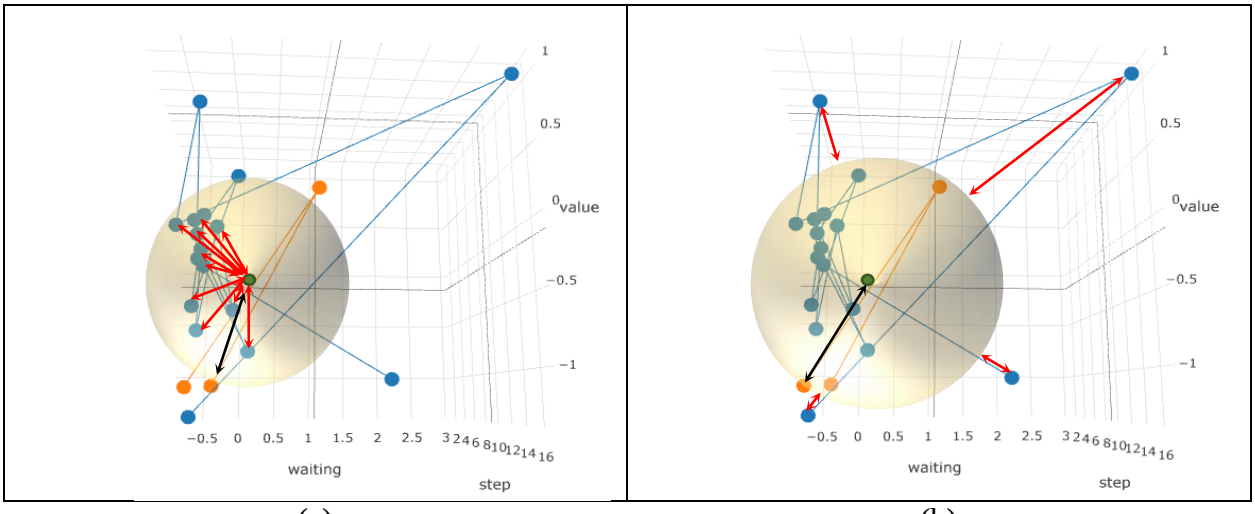
Yi lower bound (Yi-LB) can also easily be simulated in 3D DTW using the distance to the reference. Yi-LB can be defined as:

$$LB-Yi-3D(Q,C)=\sum_{i=1}^{n}\begin{cases}(Min_{j=1}^{m}(D(Q)_j)-D(C)_i)^2 & if\ D(C)_i<Min_{j=1}^{m}(D(Q)_j)\\ (D(C)_i-Max_{j=1}^{m}(D(Q)_j))^2 & if\ D(C)_i>Max_{j=1}^{m}(D(Q)_j)\\ 0 & otherwise\end{cases} \quad (4.9)$$

In Figure 19a, the red lines are the distance to reference for the points in C, which have less distance to reference than the minimum point in Q (the distance to reference of this point is shown in black line).

Also, in Figure 19b, the red lines are the distance to reference for the points in C, which have greater distance to reference than the maximum point in Q (the distance to reference of this point is shown in black line).

The summation squared of all these red lines represent the Yi lower bounding measure in 3D DTW.



(a) (b)

Figure 19. The black lines in a) and b) show the distance to reference of the minimum and the maximum points on Q, respectively and red lines in a) and b) show the distance to the reference of all points in C which are less than minimum point of Q and larger than

4.3 Early Abandoning on 3D DTW

Early abandoning in 3D DTW works similar to Early abandoning in regular DTW [26]. The objective is to eliminate the search if the accumulative 3D distances between two CCA transformed time series is greater than a predefined threshold (best-so-far). Algorithm 3.2, shows how to calculate 3D DTW distance along with early abandoning.

Algorithm 3.2 The 3D DTW algorithm with Early Abandoning

1. **function** 3D_DTW(Q=$\{(v_1^q, w_1^q), \dots, (v_i^q, w_i^q) \dots (v_1^q, w_n^q)\}$,C=$\{(\mathrm{v}_1^{\mathrm{c}}, \mathrm{w}_1^{\mathrm{c}}), \dots, (\mathrm{v}_{\mathrm{i}}^{\mathrm{c}}, \mathrm{w}_{\mathrm{i}}^{\mathrm{c}}) \dots (\mathrm{v}_{\mathrm{n}}^{\mathrm{c}}, \mathrm{w}_{\mathrm{n1}}^{\mathrm{c}})\}$, BSF: best so far (closest query))
2. $E[0,0] \leftarrow (v_i^c - v_j^q)^2 + (w_i^c - w_j^q)^2$ # Initialize first cell
3. **for** j ← 1 … m **do** # Initialize first row
4. $E[1,\mathrm{j}] \leftarrow E[1, \mathrm{j}-1] + (v_1^c - v_j^q)^2 + (w_1^c - w_j^q)^2$
5. **end for**
6. **for** i ← 1 … n **do** # Initialize first column
7. $E[\mathrm{i},1] \leftarrow E[1, \mathrm{j}-1] + (v_i^c - v_1^q)^2 + (w_i^c - w_1^q)^2$
8. **end for**
9. **for** i ← 1 … n **do**
10. **for** j ← 1 … m **do**
11. $k \leftarrow \min \begin{cases} E[i-1, j-1] \\ E[i-1, j] \\ E[i, j-1] \end{cases}$
12. $E[\mathrm{i},\mathrm{j}] \leftarrow k + (v_i^c - v_j^q)^2 + (w_i^c - w_j^q)^2$
13. **end for**
14. min_cost←min(E[i,.])
15. **if** min_cost > **BSF then**
16. return ∞
17. **end if**
18. **end for**
19. **return** E[m,n]
20. **end function**

4.4 Experiments and Results

3D DTW performance is evaluated using the UCR public benchmark datasets archive [43]. In the UCR time series classification archive, there are 85 time series from different domains and problems. Each dataset in this archive comes in two parts, a train partition and a test partition. The train partition is used in the model building process, and the test partition is used for measuring the classification accuracy. In all time-series datasets, the data is z-normalized prior to the experiment to have a mean of 0 and variance of 1.

In this experiment, we show the classification result with s=0.6 and show that even without prior knowledge about the structure of the data and suitable s candidates, 1-NN 3D DTW with a default value for *s* will work well in terms of accuracy and running time, especially on the long time series.

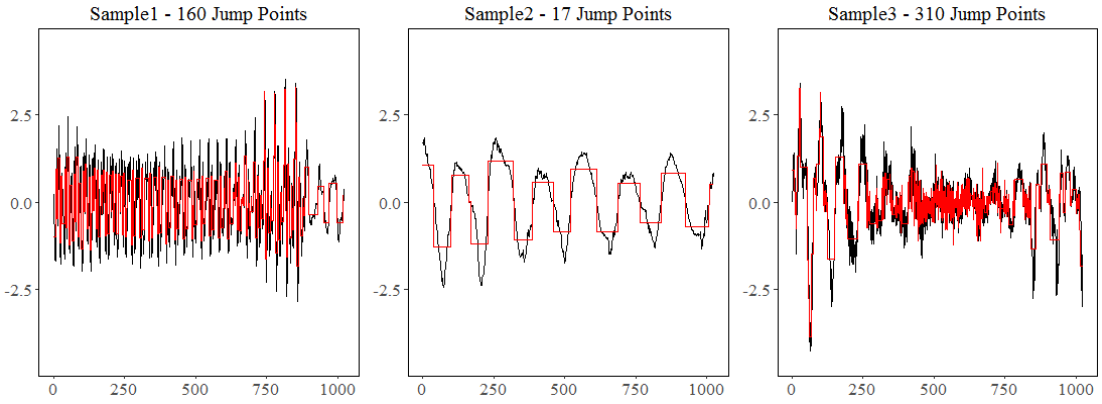
Our webpage reports all raw numbers and contains the C++ source code for 1-NN 3D DTW. All experiments were performed on an Intel Xeon E5-2620 (2.00 GHz) machine with single core setting. All C++ codes were compiled with gcc-5.3 with –O3 optimization. We will show a comparison of 1-NN 3D DTW classifier to 1-NN ED, 1-NN DTW, BOSS VS [45], Random forest, SVM with a quadratic kernel, Naive Bayes classifier, and Random Forest.

4.4.1 Case study

For the purpose of the case study, two long and large time series datasets are selected from the UCR archive. The purpose is to demonstrate the very efficient running time of 1-NN 3D DTW classification, as well as its high accuracy on long time series.

4.4.1.1 Phoneme

Phonemes are the smallest units of intelligible sound produced by a human being, and phonetic spelling is the sequence of phonemes that a word comprises. The original *phoneme* dataset is presented in [44], which has 370,000 phonemes and massive amounts of noise. It is one of the largest single-dimension time series classification dataset. The data in the original dataset are of un-equal length. Having as much as 39 classes, Phoneme time series are very challenging to classify. Three sample time series with different classes are randomly selected from the dataset, which are shown in Figure 20. The black graph shows the raw data, and the red graph shows the jump points and steps after CCA representation and data reduction. From the left graph to the right graph, respectively, 1024 points of the raw data are reduced to 160, 17 and 310 points (segments) by CCA transformation, respectively.



(a) Sample 1 (b) Sample 2 (c) Sample 3

Figure 20. Graphical presentation of 3 samples of Phoneme datasets with 3 different classes. . The raw data is presented in black line and the approximation in red line.

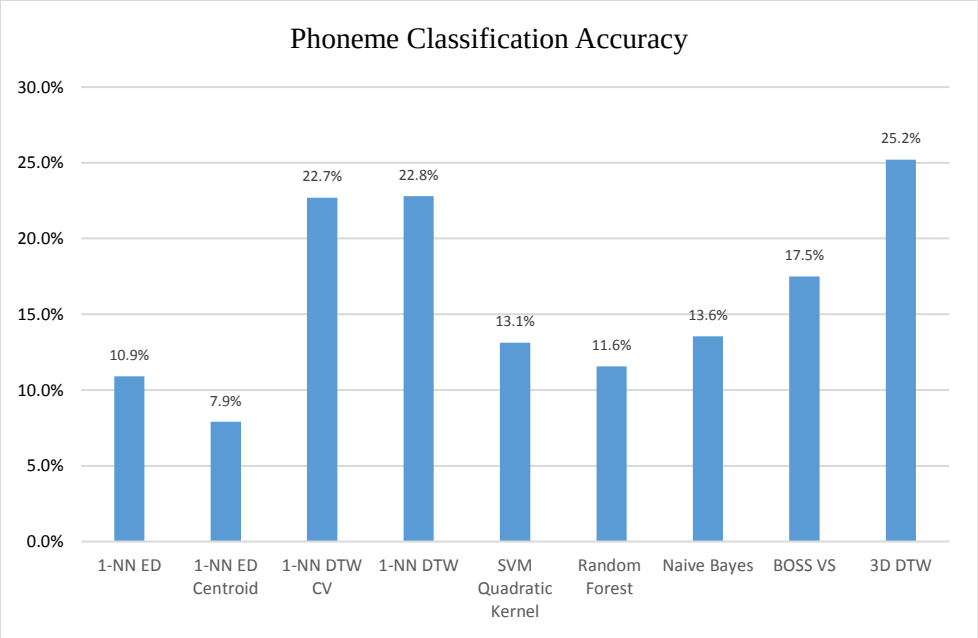


Figure 21. Phoneme classification accuracy comparison.

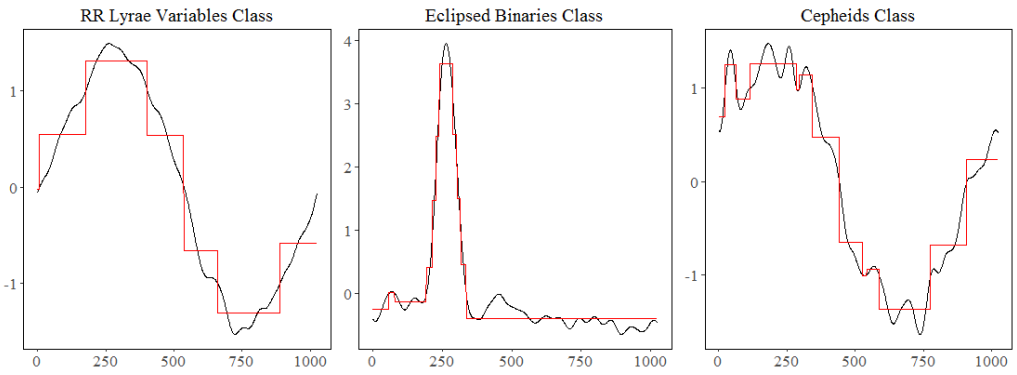
The classification accuracy and process time of 1-NN-3D DTW are compared with state-of-the-art implementation of 1-NN DTW and BOSS VS in Figure 21. 3D DTW has the highest accuracy with 25.2%. Classification of this problem by 1-NN DTW on a single core machine takes around 45 minutes, while 1-NN-3D DTW prediction is done in less than 5 minutes (close to BOSS VS) with better accuracy.

4.4.1.2 Starlight curves

StarLight curves is the largest dataset available in the UCR time series archive [43]. It contains 1000 train and 8236 test records with a length of 1024 in three types of classes (star objects): Eclipsed Binaries, Cepheid and RR Lyrae Variables.

Figure 22, shows sample graphs in each class. The difficult part of classification is detecting RR Lyrae Variables from Cepheids due to their similar shapes. Since there is smooth change of values in this problem, the CCA representation is very effective, and it can significantly reduce the number of points after transformation. This will result faster classification process.

Figure 22, shows, from left to right, reduction of 1024 raw data points to 7, 19 and 12 points (segments) by CCA transformation respectively. Figure 23, compares the classification accuracy among classifiers. 1-NN 3D DTW accuracy is among the best scores and nearly the same as 1-NN DTW while significantly outperforming every other classifier in time.



(a) RR Lyrae Variables (b) Eclipsed Binaries (c) Cepheids

Figure 22. Graphical presentation of 3 samples of Starlight Curves datasets with 3 different classes. The raw data is presented in black line and the approximation in red line.

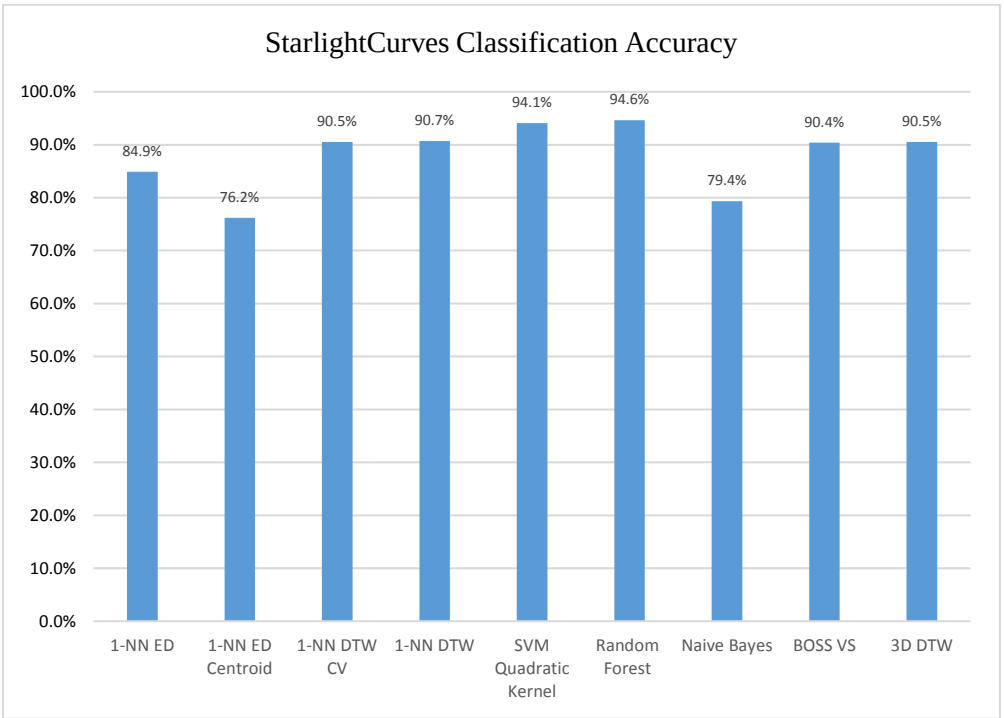


Figure 23. Starlight curves classification accuracy comparison.

While the state-of-the-art 1-NN DTW takes 2 hours, and BOSS VS takes 3 minutes to process the classification of this problem, 1NN-3D DTW does the same job in less than 8 seconds on the same machine with the same accuracy. Whereas BOSS VS is among the fastest classifiers and is significantly faster than 1-NN DTW CV, 1-NN DTW, SVM and Random Forest [45], it is outperformed by 1NN 3D DTW by about 20 times.

4.4.2 Classification Accuracy

Figure 24, shows the comparison of 1-NN 3D DTW to 1-NN DTW classification accuracy for all datasets in the UCR archive. 1-NN DTW is frequently used as the benchmark for comparison [6]-[9]. In this figure, each point represents one dataset. The x-axis is the difference between 1-NN 3D DTW and 1-NN DTW classification accuracies, and the y-axis is the length of time series. The points on the right side of the y- axis are the datasets which 1-NN 3D DTW performs better, and the points on the left side of the y-axis are the datasets which 1-NN DTW does better in terms of classification accuracy.

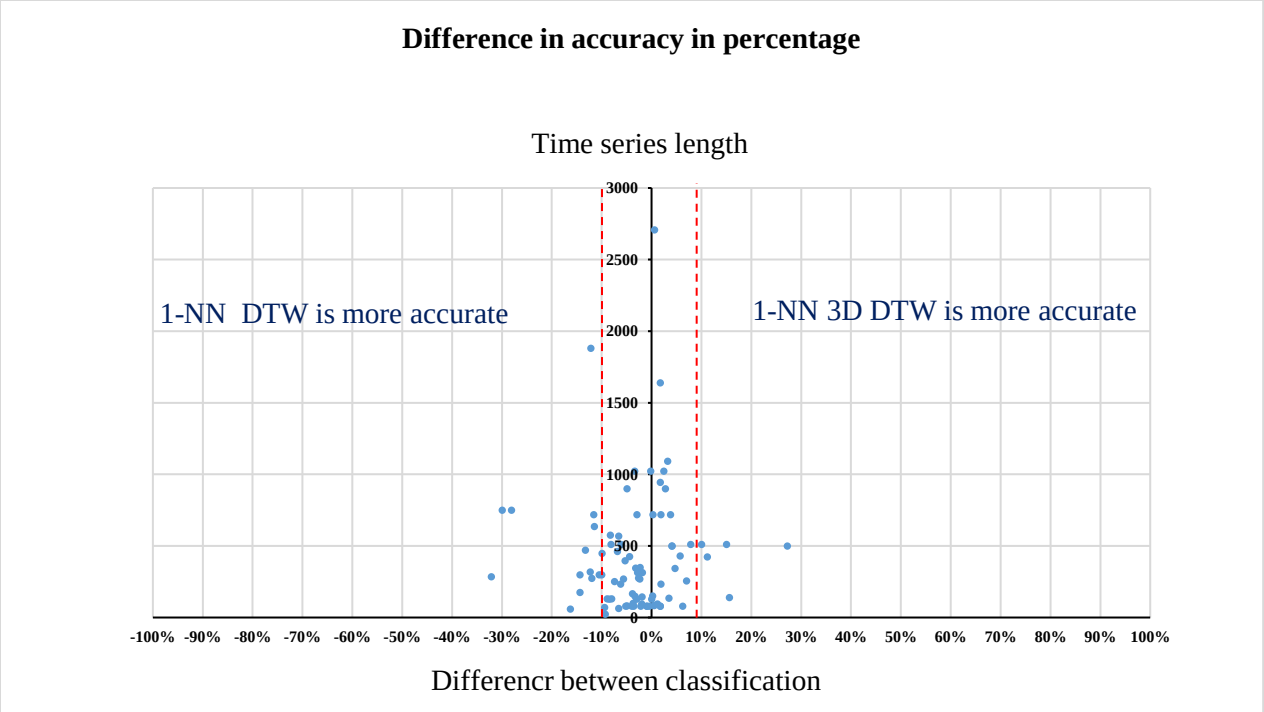


Figure 24. Difference in accuracy between 1-NN 3D DTW and 1-NN DTW classification on all 85 time series datasets in UCR archive with different time series lengths.

Figure 24, shows that 1-NN 3D DTW performance in term of accuracy is competitive to 1-NN DTW. The majority of datasets fall within a range of -10% to 10%. However, the main advantage of our method is in accelerating the classification process rather than improving the accuracy.

The classification time becomes a more significant factor for long time series, and it is where most current classifiers fall short. In many use cases a faster prediction is much more important than a non-significant increase in accuracy. For that reason, we are interested to see how our method works on long time series (500 points and above). There are 28 datasets in UCR archive with more than or equal to 500 lengths. Table III, shows the list of these datasets along with their lengths.

TABLE III. THE LIST OF LONG TIME SERIES DATASETS IN URC ARCHIVE

Dataset name	**Length**	**Dataset name**	**Length**
BeetleFly	512	MALLAT	1024
BirdChicken	512	NonInvasiveFatalECG1	750
Car	577	NonInvasiveFatalECG2	750
CinC_ECG_torso	1639	OliveOil	570
Computers	720	Phoneme	1024
Earthquakes	512	RefrigerationDevices	720
FordA	500	ScreenType	720
FordB	500	ShapeletSim	500
HandOutlines	2709	ShapesAll	512
Haptics	1092	SmallKitchenAppliances	720
Herring	512	StarlightCurves	1024
InlineSkate	1882	UWaveGestureLibraryAll	945
LargeKitchenAppliances	720	Worms	900
Lighting2	637	WormsTwoClass	900

First, we compare the accuracy of 1-NN 3D DTW on long time series with 1-NN ED and 1-NN DTW which are the most common time series classifiers. We also review the accuracy of 1-NN ED Centroid and 1-NN DTW Centroid which are considered to be very fast classifiers.

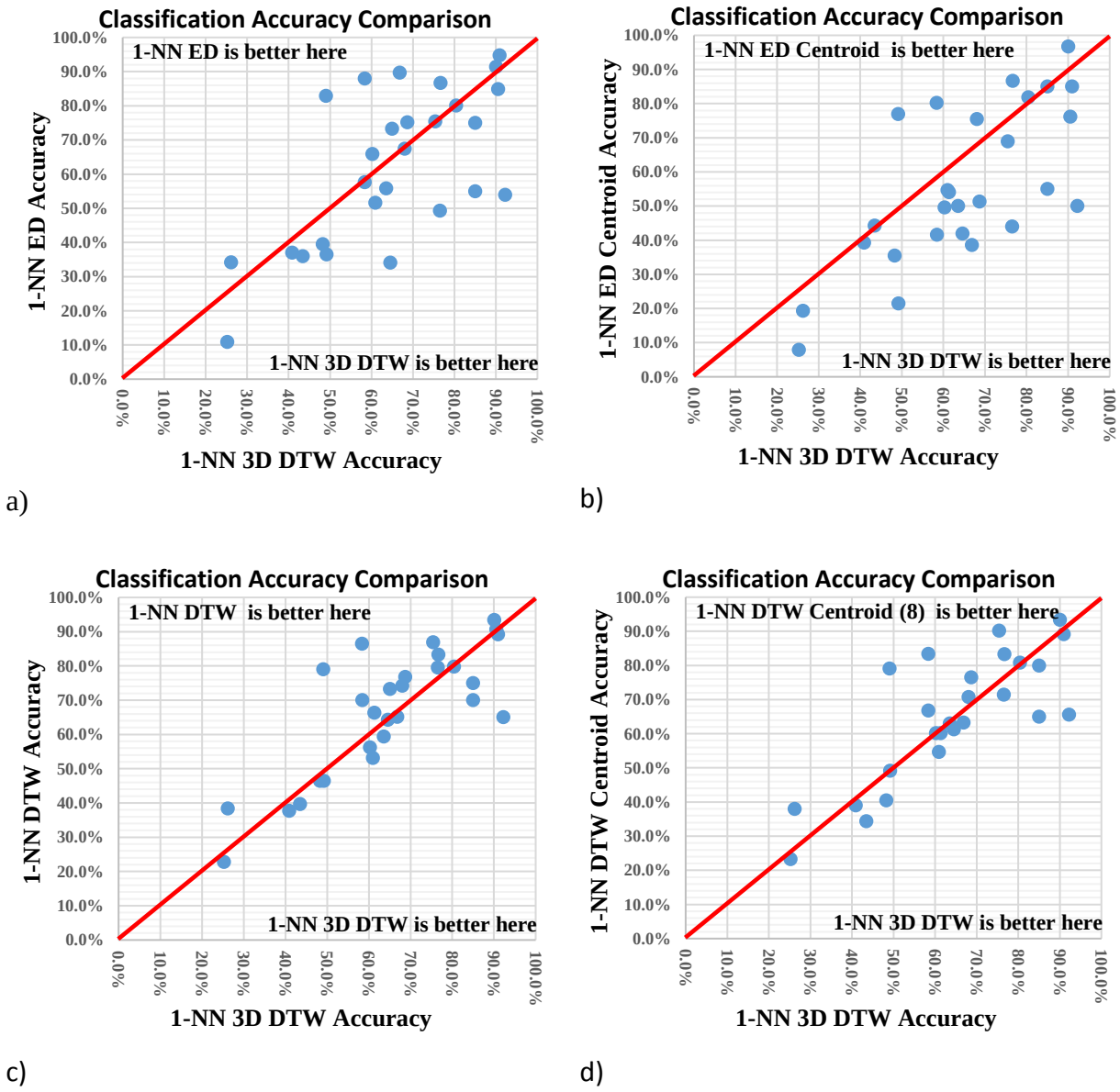


Figure 25. Pairwise classification accuracy comparison of 1-NN 3D DTW with 1-NN ED, 1-NN ED Centroid, 1-NN DTW and 1-NN DTW Centroid (8) on all long datasets

Figure 25, shows pairwise classification accuracies comparison between 1-NN 3D DTW and (a) 1-NN ED, (b) 1-NN ED Centroid, (c) 1-NN DTW and (d) 1-NN DTW Centroid. Each long time series dataset is represented by one blue point. Each point under the straight red line indicates that 1-NN 3D DTW is more accurate. The 1-NN 3D DTW offers a significantly higher accuracy

than 1-NN ED, 1-NN ED Centroid and 1-NN DTW Centroid. Our 1-NN 3D DTW also outperforms the accuracy of 1-NN DTW for more than half of long time series datasets (15 win/13 lose). In the next section, we show that 1-NN 3D DTW is orders of magnitude faster than 1-NN DTW.

Figure 26, shows the pairwise accuracy comparison of 1-NN 3D DTW with Naïve base (a fast classifier), SVM (with a quadratic kernel), Random Forest (which is a tree based ensemble method) and the state-of-the-art BOSS VS classifier [45].

The 1-NN 3D DTW classifier outperforms Naïve base, Random Forest, and SVM. On most long time series datasets, the accuracy difference of 1-NN 3D DTW to BOSS VS is within -15% to 10%. Note that there are many circumstances in which we would prefer to sacrifice some levels of accuracy for considerable speedup [31] [41]. The advantage of the 1-NN 3D DTW classifier is its efficient running time before its classification accuracy.

Note that here we did not compare the accuracy of ensemble methods such as COTE [46], Elastic Ensemble (PROP) [7], BOSS [47] and Shotgun [48]. They might improve the accuracy but their outstanding accuracy is achieved at the cost of a very high classification time. For example, the COTE classifier is an ensemble of 35 different classifiers, and it has to run all of these classifiers first to predict a label.

We also did not compare the accuracy of TSBF [49], Fast Shapelets [50], Logical Shapelets [51], Bag-of-Patterns [52] and SAX-VSM [53] because BOSS VS classifier outperforms significantly these classifiers significantly in time or accuracy.

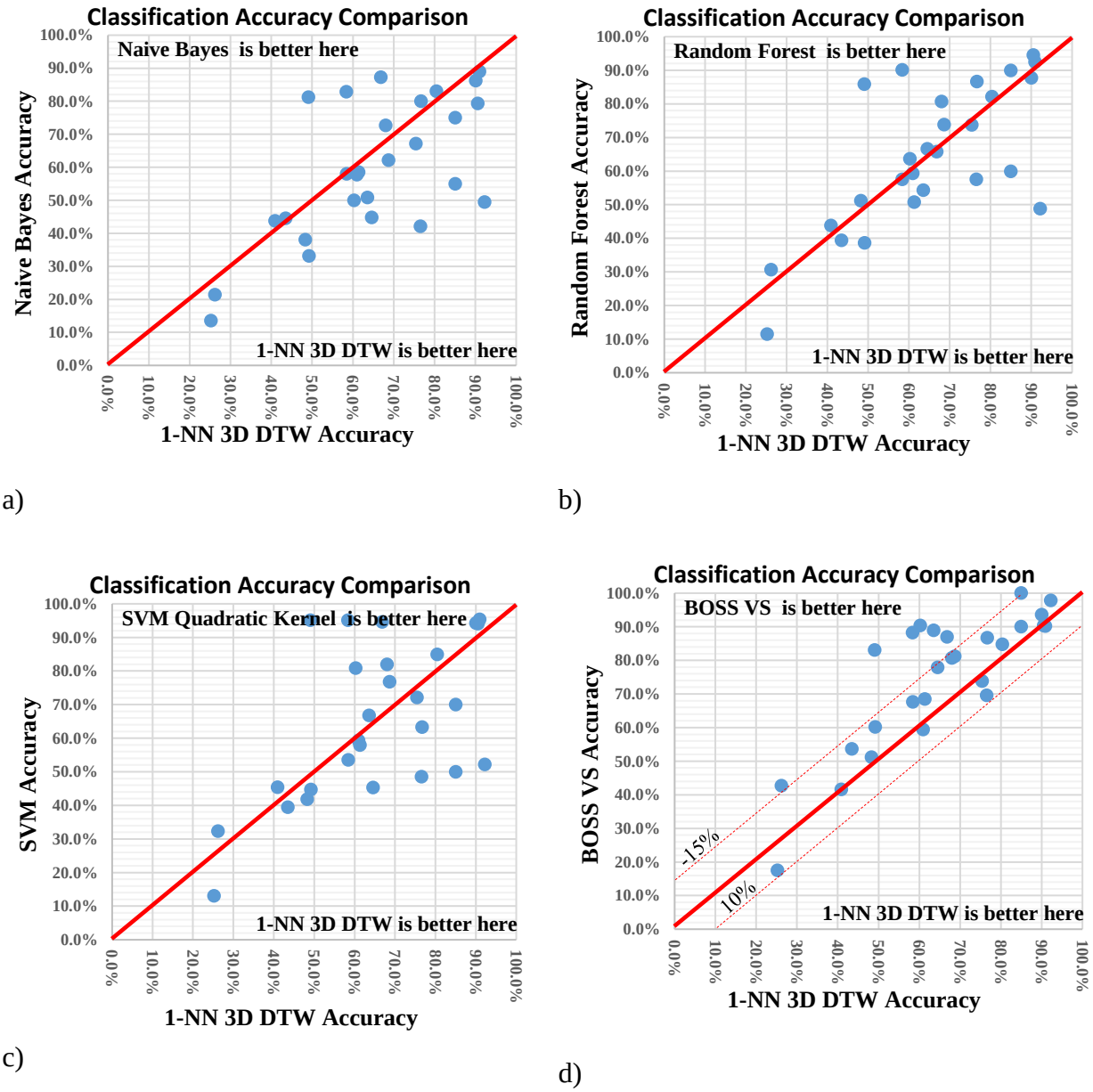


Figure 26. Pairwise classification accuracy comparison of 1-NN 3D DTW with Naïve Bayes, Random Forest, SVM Quadratic Kernel and BOSS VS on all long datasets.

4.4.3 Classification Speed

Figure 27, demonstrates the pairwise comparison of classification time of 1-NN 3D DTW with 1-NN DTW on all 85 time series in URC archive. Each point represents a classification time

(in seconds) of one time series dataset. The points above the straight blue line indicate less classification runtime in 1-NN 3D DTW compared to 1-NN DTW.

As seen in Figure 27, 1-NN 3D DTW has a huge performance leap over 1-NN DTW in terms of time on all UCR time series datasets. It takes only around 45 minutes to classify all 85 time series datasets by the 1-NN 3D DTW using 1 CPU core, whereas the same job takes more than 18 hours by the state-of-the-art 1-NN DTW implementation and also more than 1.2 hours by a single-core running BOSS VS classifier.

Using 1-NN 3D DTW, 73 out of 85 datasets (86%) are classified in less than 10 seconds, and 80 out of 85 datasets (92%) are classified less than 1 minute. For the long time series datasets, using 1-NN 3D DTW, the classification of all datasets finishes roughly in 37 minutes.

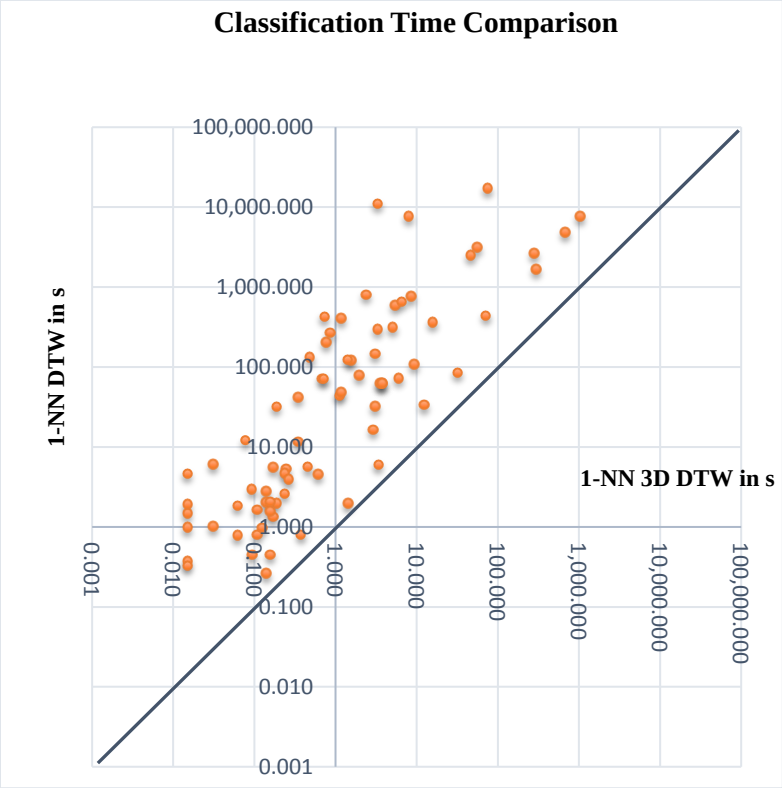


Figure 27. Pairwise classification time comparison of 1-NN 3D DTW with 1-NN DTW on all 85 time series datasets.

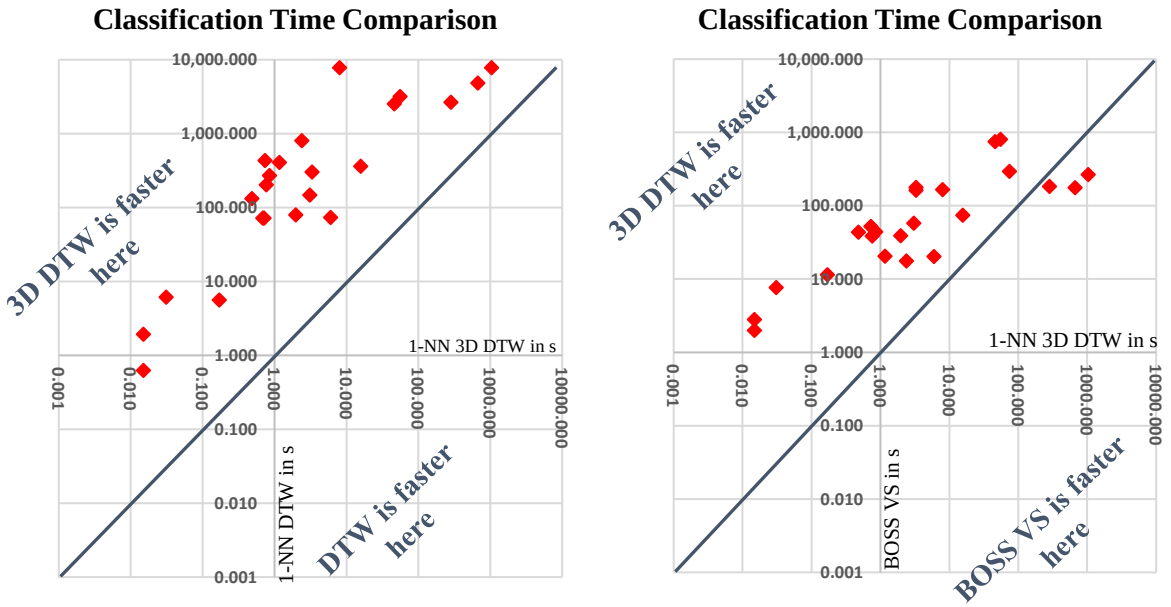


Figure 28. Pairwise classification time comparison of 1-NN 3D DTW with 1-NN DTW and BOSS VS on all long datasets

Using 1-NN DTW and BOSS VS the classification finishes approximately after 17 hours and 1 hour, respectively.

On the trade-off between classification accuracy against processing time and computational cost, 1-NN 3D DTW performs reasonably well. On one hand, it is considerably more accurate than extremely fast classifiers (such as 1-NN ED, 1-NN ED Centroid, Naïve Bayes, etc.), and, on the other hand, it is significantly faster than extremely accurate classifiers (such as COTE Ensemble, Elastic Ensemble PROP, BOSS Ensemble, Shotgun Ensemble, etc.).

The details of classification time comparison between 1-NN 3D DTW, 1-NN DTW, and BOSS VS on all long time series datasets are presented in Figure 28, 1-NN 3D DTW is significantly faster than 1-NN DTW with similar or better accuracy. It is orders of magnitude faster than 1-NN DTW in all long time series.

1-NN 3D DTW is faster than BOSS VS with a slightly declined accuracy. In all the long time series datasets, 1-NN 3D DTW is faster than BOSS VS except only three time series datasets (*Phoneme*, *FordA* and *FordB*).

4.4.4 Texas Sharpshooter Plot

Along with measuring the accuracy and time classification of 1-NN 3D DTW and comparing it to other methods, it is needed to predict ahead of time on which datasets and domains 1-NN 3D DTW will reach superior classification accuracy results. The Texas sharpshooter plot [10] is the tool to predict if one method or the other is performing better in terms of classification accuracy. The goal is to predict the test classification accuracy for 1-NN 3D DTW and the 1-NN ED based on the classification accuracy on the training datasets.

In order to create the Texas sharpshooter plot, we used the expected gain equation which is proposed in [10],[11].

$$\text{Expected gain} = \frac{1 - \text{NN 3D DTW accuracy}}{1 - \text{NN ED accuracy}}$$

We used 3- fold cross validation to calculate the classification accuracy on the train datasets. If the expected gain is greater than 1, we can assume that 1-NN 3D DTW will have a better result than 1-NN ED in the testing of that particular dataset too. In Figure 29, the comparison of the actual gain (based on the testing dataset) against the expected gain (based on the training dataset) of 1-NN 3D DTW and 1-NN ED is presented.

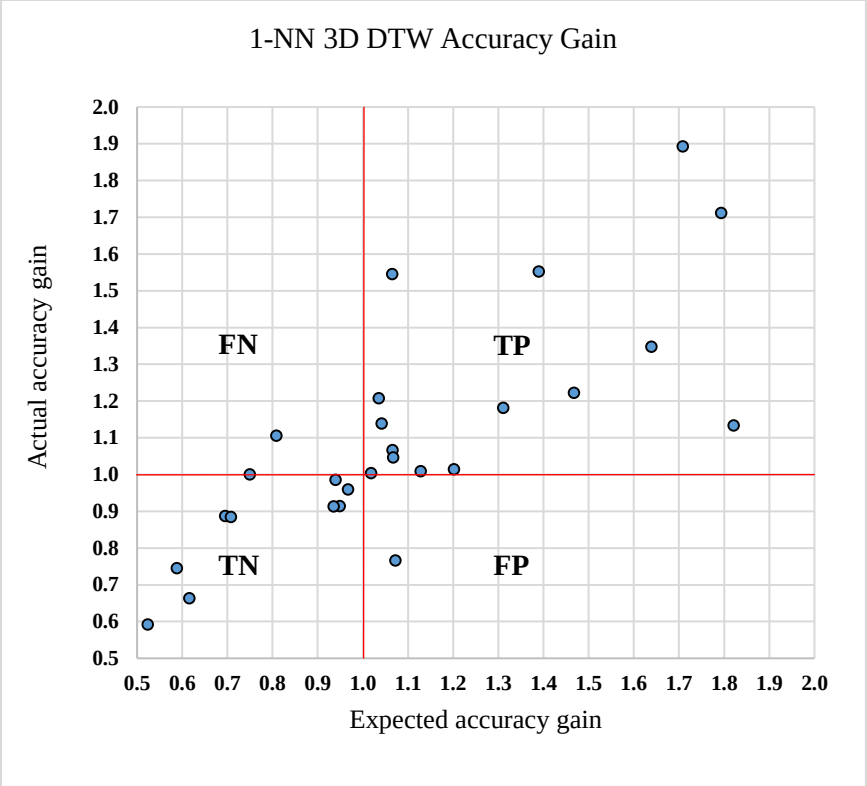


Figure 29. Expected accuracy gain of 1-NN 3D DTW/1-NN ED from train data compared to actual accuracy gain on test data for long datasets.

There are 4 areas in this plot:

– True Positive (TP): Where expected gain is greater than 1 and actual gain is also greater than 1. It means we expected the better accuracy and we received the better accuracy. 16 out of 28 datasets fall into this area.

– False Negative (FN): Where expected gain is less than 1, but the actual gain is greater than 1. It means we expected worst accuracy, but we got the better accuracy. 1 out of 28 datasets falls into this area (*Haptics*).

– True Negative (TN): Where expected gain and actual gain are both less than 1. It means we correctly expected that the accuracy would decrease. 10 out of 28 datasets fall into this area.

– False Positive (FP): Where expected gain is greater than 1, but the actual gain is less than 1. It means we wrongly expected the accuracy to improve, but it got worse. This is the bad area because we decided to use 1-NN 3D DTW based on the expected gain on the training but we lost accuracy in testing. There is only 1 out of 28 datasets which falls into this area (*InlineSkate*).

Therefore, in general 1-NN 3D DTW enjoys a high level of predictability for the classification accuracy of long time series datasets.

4.5 Finding Optimal Resolution Parameter

The resolution of reduction is parameterized (*s*), and can be used to control the coarsening of the transformation, which influences the running time and accuracy. One challenge by applying the CCA is choosing the proper *s* for approximation. Our experiments with various range of *s* values over all the UCR datasets, could not verify a meaningful trend of accuracy relative to *s* consistent with all the datasets. This parameter can be selected according to the prior knowledge of the data, or by cross-validation. Cross validation can be efficient and effective when improving accuracy is of higher concern.

The results showed that, on most time series, choosing an *s* value, which creates an even number of partitions (with the middle threshold falling on zero line), generally obtain better results in terms of accuracy. On a large scale, some particular s values were observed to perform better on average. In Figure 30, the effect of *s* values on the classification error rate for some example datasets is presented.

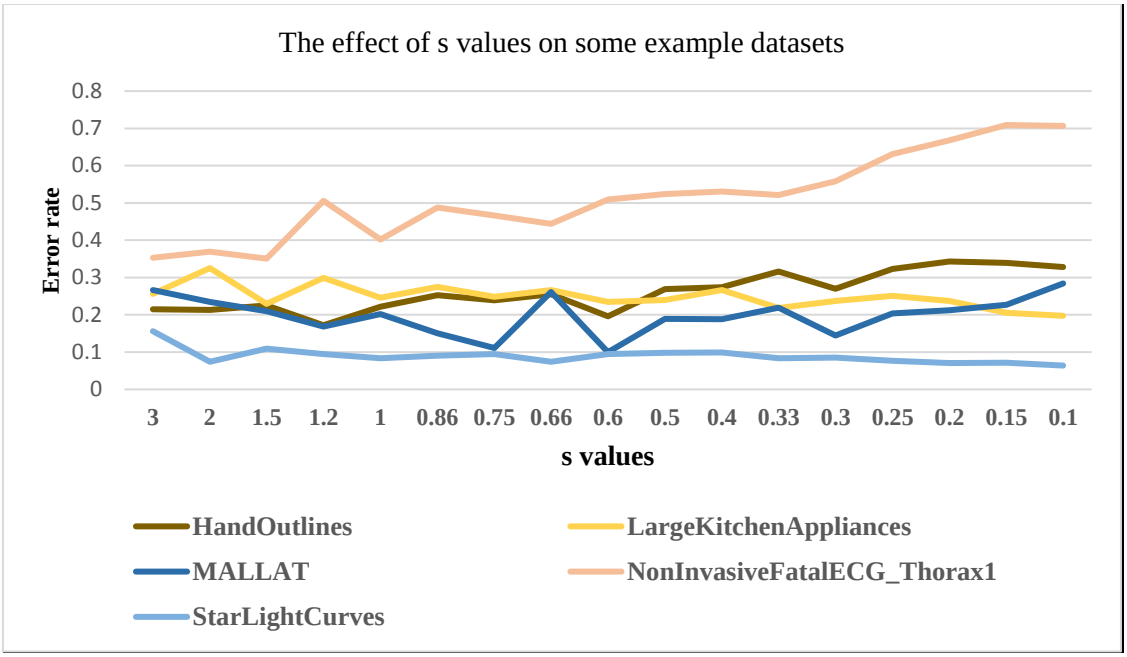


Figure 30. The effect of s values on some example datasets

5 BLOCKED DYNAMIC TIME WARPING

Material from my own previously work [75], 2017 KAIS, is reused in this chapter.

5.1 Introduction

There are two approaches to deal with computational complexity of DTW (which is its main downside): Using approximation of DTW and utilizing pruning strategies in the exact DTW calculation.

Diverse estimate and approximation strategies are produced to make DTW quicker while keeping its accuracy (precision) level. Utilizing constrained warping window in DTW matrix and applying DTW on reduced-dimension representation of the time series are examples of these approximation techniques.

In 2016, AWarp (Warping Distance for Sparse Time Series) was presented which is viewed as another effective approximation technique. However, this technique is appropriate only for sparse time series. Sparse time series are an special type of time series which contain repetition of zeros alongside non-zero values [65].

AWarp exploits repetition (runs) of zeros in time series, however there are many cases that repeated values in time series are non-zero. AWarp is not valuable any longer in these cases. Figure 31, demonstrates a few cases of time series with no-zero values repetition from UCR time series archive [43].

In this Chapter, Blocked Dynamic Time Warping (BDTW) is exhibited as another warping distance alongside its variations which deal with time series with runs of any values (zeros or non zeros).

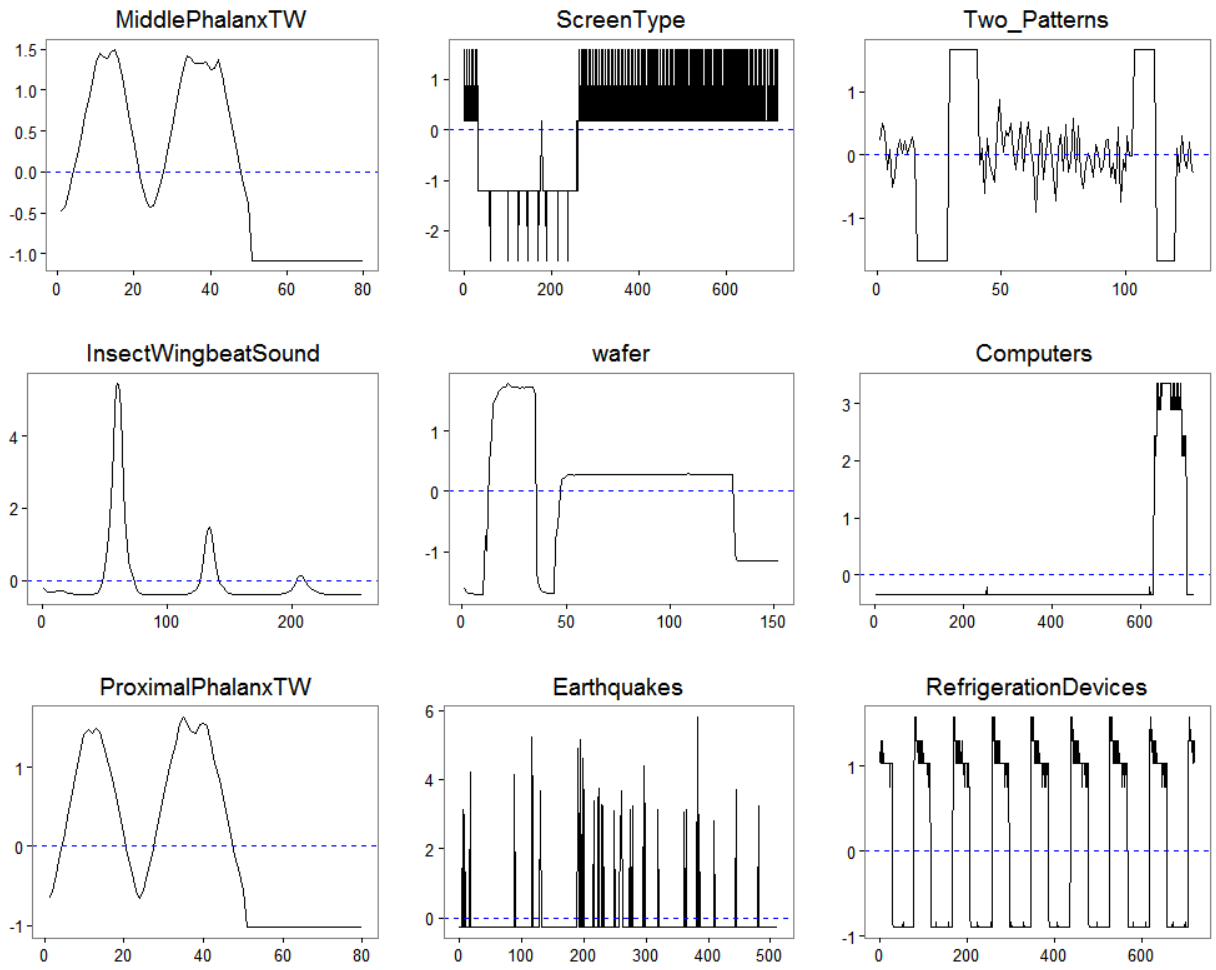


Figure 31. Example of time series with values repetition

Before using BDTW, time series should be Run Lengths encoded. Run Lengths encoding uses the repetition of values in time series and reduces the lengths (dimension) of time series. This is the reason which makes BDTW calculation faster than calculation of original DTW (based on the original length of time series). For binary-valued (and in fact any two-valued) time series with runs (repetition) of zero or non-zero values based on our experiments, BDTW results in the exact DTW distance. BDTW also obtains a lower bound and an upper bound for any-valued time series which has repetition of any values (zero or non-zero).

BDTW_UB, BDTW upper bound distance, is a close approximation of DTW distance. The more value repetition rate of a time series causes the more shrinkage of time series length after Run Lengths encoding. Therefore, when using BDTW, the higher value repetition rate of time series, causes the shorter processing time. Because the processing time depend on the length of time series.

We can also combine BDTW with several time series representation methods such as PAA and APCA, to obtain a new approximation method which is applicable for time series with low rate of value repetitions or even without any value repetitions.

5.2 Related Work

5.2.1 Warping Distance for Sparse Time Series

The AWarp method was presented in 2016 as a new DTW approximation technique for sparse time series. Sparse time series include disproportionately-spread runs of zeroes and none-zero values as observations. For sparse time series, AWarp introduces a lower bound and an upper bound of DTW distance. However, AWarp only woks on sparse time series and only takes advantage of repetition of zero but our proposed method, BDTW, utilizes repetition/runs of any values.

The comparison of utilization of AWarp and BDTW is summarized in Table IV. AWarp and BDTW both (based on our experiments) result in the exact DTW distance for binary-valued time series. If a binary-valued time series has both repetition of zeros and ones, in AWarp, repetition of ones is not useful in reducing the calculation time of the exact DTW. However, BDTW exploits both the repetition (runs) of zeros and ones. In addition to binary-valued time series, any two-valued time series can be used with BDTW which again results in the exact DTW.

AWarp is helpful for more-than-two-valued time series with repetition (runs) of zeros but, BDTW is usable for more-than-two-valued time with repetition (runs) of any values. Combination of BDTW and APCA obtains an approximation method which can be used for all time series with or without value repetition.

TABLE IV. THE COMPARISON OF THE APPLICATION OF AWARP AND BDTW

Time series type	**Usable method**		**Output**
Two valued- binary-with repetition of zeros	AWarp	BDTW	exact DTW
Two valued- binary-with repetition of ones	-	BDTW	exact DTW
Two valued-non binary-with repetition of any values	-	BDTW	exact DTW
More than two valued-with repetition of zeros	AWarp	BDTW	DTW upper and lower bounds
More than two valued-with repetition of any values	-	BDTW	DTW upper and lower bounds
More than two valued- without any values repetition	-	BDTW+APCA	DTW approximation

BDTW has another benefit over AWarp. BDTW is compatible with z-normalization of time series. After z-normalization of a sparse time series with repetition of zeros, zeros will change to non-zero values and AWarp would not be helpful anymore. One the other hand, BDTW in this case is still applicable after z-normalization because it works with repetition of none-zero values as well. This is an important issue since usually z-normalization is an essential stage in preprocessing for classification of time series to evade some invariance of time series such as amplitude and offset invariances.

5.2.2 Using an Upper Bound to Speed up All-Pairwise DTW Matrix Calculation

In 2016, PrunedDTW was introduced as a method to accelerate the calculation of exact DTW matrix [66]. PrunedDTW can be used in all-pairwise DTW calculation which is required in classification and clustering of time series. In PrunedDTW, Euclidean Distance (ED) is used as an

upper bound for pruning the cells inside DTW matrix which are considered as unhopeful alignments. The main idea of PrunedDTW comes from a basic observation: in each DTW matrix, a cell with relatively great value typically is not likely to be on the optimal pass. By considering a DTW upper bound and finding the cells which have higher than that upper bound, we can detect the other cells which have no chance of being on the optimal path and we can prune these cells in the process of DTW calculation and filling of DTW matrix.

In one hand since the time complexity of ED is linear it does not impose much of time overload on DTW calculation. But on the other hand it is not a tight upper bound and its pruning power in DTW calculation is limited.

5.2.3 PAA Based DTW Approximation

Several approximation methods have been developed to address the problem of high time complexity of DTW [67-70,74]. One of these methods is based on combination of PAA and DTW [67]. This method is called Abstraction. The abstraction method is performed in 3 steps:

1- Transforming time series to constant segments and reducing the resolution of time series using PAA.
2- Applying DTW on a reduced resolution of time series and finding the optimal path on the abstraction level of the time series.
3- Mapping the optimal path on the original time series and approximating the DTW distance.

5.3 Blocked Dynamic Time Warping

5.3.1 Time Series Encoding

A time series Run Lengths encoding is defined by considering the repetitive values and their number of runs (the quantity of their repetition). In this encoding each repetitive value is

represented one time for each run along with its number of runs inside a parenthesis. This encoding reduces the length of time series with repetitive values. For instance, a time series T= {1,3,3,3,3,2,2,2,3,3,3,3,3,1,1,2,2,2,2,1}, after encoding is presented as T_{ENC}={1(1),3(4),2(3),3(5),1(2),2(4),1(1)}. In this example, we can see that the original length of time series after encoding changes from 20 to 7. Since in DTW algorithm the first values of two time series must be aligned together as well as the last values of time series, we should keep the first value and last value of the time series as single values (with one repetition). In order to do that if a time series starts with repetition (runs) of a value or ends with repetition (runs) of a value, we keep the first value (and last value) as one repetition and encode the rest of values. For instance, a time series A = {3,3,3,3,3,3,1,1,1,2,2,2,2,2,2,2}, after encoding is presented as A_{enc} ={3(1),3(5),1(3),2(6),2(1)}.

Next, a measure is defined that represents the level of repetitiveness of time series. This measure is named repetitiveness rate (*rr*) or repetition rate. The repetitiveness rate indicates how frequent values are repeated in a time series. If the original length of a time series is l and the length of time series after encoding is l', the time series repetitiveness rate (*trr*) is defined as:

$$trr = \left(1 - \frac{l'}{l}\right) * 100 \qquad (5.1)$$

The repetitiveness rate (rr) can also be defined for a dataset. For a time series dataset with *K* instances (that i^{th} time series has the length of l_i and *i* is from 1 to *K*), the repetitiveness rate (rr) is defined as:

$$rr = \left(1 - \frac{\sum_{i=1}^{K} l'_i}{\sum_{i=1}^{K} l_i}\right) * 100 \qquad (5.2)$$

If in a time series dataset, original length of all instances are equal (with length of *n*), the repetitiveness rate is defined as:

$$rr = \left(1 - \frac{\sum_{i=1}^{K} l'_i}{Kn}\right) * 100 \qquad (5.3)$$

A higher rr of a dataset, shows higher level of values repetition in that dataset.

5.3.2 Blocked Dynamic Time Warping Structure

Definition

In DTW matrix, each cell represents the alignment of values between two time series. Set of adjacent cells that represent the alignments of repetitive values in a DTW matrix is named a **block**. In Figure 32a, an example traditional DTW matrix is shown and blocks are depicted with different colors in this matrix. In Figure 32b, a Blocked Dynamic Time Warping matrix is presented and each cell in this matrix corresponds with a block in the conventional DTW matrix.

	1	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**
0	**1**	1	**1**	**2**	**2**	3	4	**5**	**5**
1	1	2	2	1	2	2	2	2	3
1	1	2	3	1	2	2	2	2	3
1	1	2	3	1	2	2	2	2	3
1	**1**	2	**3**	**1**	**2**	2	2	**2**	**3**
0	2	1	**1**	**2**	**1**	2	3	**3**	**2**
1	2	2	**2**	**1**	**2**	1	1	**1**	**2**
0	3	2	2	2	1	2	2	2	1
0	4	2	2	3	1	2	3	3	1
0	**5**	2	**2**	**3**	**1**	2	3	**4**	**1**
1	**5**	3	**3**	**2**	**2**	1	1	**1**	**2**

a) Traditional DTW matrix

	1(1)	**0(2)**	**1**	**0**	**1(3)**	**0(1)**
0(1)	**1**	**1**	**2**	**2**	**5**	**5**
1(4)	**1**	**3**	**1**	**2**	**2**	**3**
0(1)	**2**	**1**	**2**	**1**	**3**	**2**
1(1)	**2**	**2**	**1**	**2**	**1**	**2**
0(3)	**5**	**2**	**3**	**1**	**4**	**1**
1(1)	**5**	**3**	**2**	**2**	**1**	**2**

a) Blocked DTW matrix

Figure 32. Traditional DTW matrix and Block DTW matrix for two binary time series (X and Y)

The Blocked Dynamic Time Warping (BDTW) works on encoded time series. The input of BDTW algorithm (Algorithm 5.1) is two encoded time series and the optimal path is calculated based on them. Since for time series with repetition of values the length of encoded time series are less than length of original time series, the computational complexity of BDTW is less than traditional DTW. Given l_x and l_y as the length of two time series, the traditional matrix will be a l_x by l_y matrix and a block in this matrix is represented by a cell in BDTW matrix. Assume value a is repeated A times in time series X and value b is repeated B time in time series Y, we would have a (A) as a point in encoded X and b (B) as a point in

encoded Y. The intersection of these two points in BDTW matrix represents the alignments within a block in the original DTW matrix.

The structure of the BDTW matrix is presented in Figure 33. In this figure, the accumulative distance of the alignments between a (A) and b (B) is $D_{i,j}$.

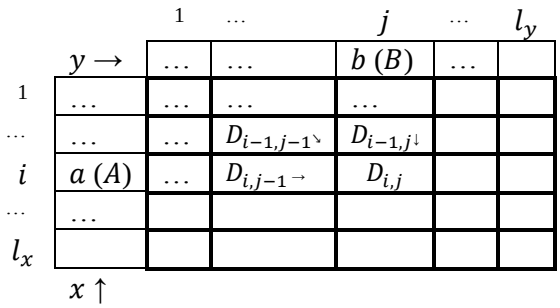


Figure 33. The structure of the BDTW cost matrix

$D_{i,j}$ is calculated based on the Euclidean distance between a (A) and b (B) points, plus the minimum accumulative distance from previous cells which are from adjacent top cell ($D_{i-1,j}$), adjacent left cell ($D_{i,j-1}$) or adjacent diagonal cells ($D_{i-1,j-1}$). The distance between a (A) and b (B) points depends on the direction of the optimal path that enters the intersection cell. If the optimal path enters the cell $D_{i,j}$, from the top cell ($D_{i-1,j}$), the distance of block is equal to $B * (a-b)^2$. If it enters from left cell ($D_{i-1,j}$), the distance of the block will be equal to $A * (a-b)^2$ and if enters from the diagonal cell ($D_{i-1,j-1}$), the distance of the block will be equal to $Max(A,B) * (a-b)^2$.

Finally $D_{i,j}$ is calculated based on the path which obtains the minimum distance:

$$D_{i,j} = \min\{\underbrace{D_{i-1,j} + \mathrm{A} * (a-b)^2}_{\text{Top}}, \quad \underbrace{D_{i,j-1} + B * (a-b)^2}_{\text{Left}}, \quad \underbrace{D_{i-1,j-1} + \max(A,B) * (a-b)^2}_{\text{Diagonal}}\}$$

5.3.3 Using BDTW for DTW Calculation of Different Time Series

5.3.3.1 BDTW on Binary-Valued Time Series

For binary (or any two-valued) time series, based on our experiments BDTW distance result is exactly same as DTW distance.

Example: Given original time series X={0,1,1,1,1,0,1,0,0,0,1} with length of 11, and Y={1,0,0,1,0,1,1,1,0} with length of 9, begin by the transforming X and Y to encoded time series: X_{ENC}={0(1),1(4),0(1),1(1),0(3),1(1)} and Y_{ENC}={1(1),0(2),1,0,1(3),0(1)} reducing both original time series to a length of 6. The encoded time series then fill the outer shell of the BDTW cost matrix. Next, apply the BDTW algorithm, the resulting matrix is represented in Figure 32b.

The conventional DTW cost matrix is presented in Figure 32a. The values in the BDTW cost matrix are identical to the values of the corresponding blocks of the DTW matrix.

Algorithm 5.1 Blocked_DTW_UB(series1, series2)

Required: series1 & series2 ←reduced time series in a 2xn data frame. The first row of the data frame is the values of the reduced time series, the second row is the number of repetitions.
Note: *A* and *B* represent the values of the points in the data reduced series, *a* and *b* are the number of times these values are repeated.
Ensure: Output the distance *d* between series1 and series2

1: $l_a \leftarrow$ length(series1), $l_b \leftarrow$ length(series2)
2: $D(1:l_a, 1:l_b) \leftarrow \infty$
3: $D_{1,1} \leftarrow (A_1\text{-}B_1)^2$
4: **for** i ← 1 to l_a
5: $D_{i,1} \leftarrow A_i(a_i\text{-}b_1)^2$
6: **end for**
7: **for** j ← 1 to l_b
8: $D_{1,j} \leftarrow B_j(a_1\text{-}b_j)^2$
9: **end for**
10: **for** i ← 2 to l_a
11: **for** j ← 2 to l_b
12: top ← $D_{i-1,j} + A_i(a_i\text{-}b_j)^2$
13: diagonal ← $D_{i-1,j-1} + \max(A_i, B_j)(a_i\text{-}b_j)^2$
14: left ← $D_{i,j-1} + B_j(a_i\text{-}b_j)^2$
15: $D_{i,j} \leftarrow$ min(top, diagonal, left)
16: **end for**
17: **end for**
18: **return** $D_{la,lb}$

5.3.3.2 BDTW on any-valued repetitive time series

For two–valued time series (including binary-valued time series), in calculation of optimal path and distance in a block, the role of all repetitive values are identical and the paths do not come from the middle of repetitive values. However, it is not the case for more-than-two-valued repetitive time series. For example if a point in one encoded time series is 4(3), all (3 times repeated) values of 4, do not play the same role in the distance calculation of the related block. A value in the middle of run (repetitions) might change the direction of the path inside the block.

	1	**3**	**3**	**3**	**2**
1	**0**	4	8	**12**	**13**
2	**1**	1	2	**3**	**3**
1	**1**	5	5	**6**	**4**
2	**2**	2	3	4	**4**

a) Traditional DTW matrix

	1(1)	**3(3)**	**2(1)**
1(1)	**0**	**12**	**13**
2(1)	**1**	**3**	**3**
1(1)	**1**	7	**4**
2(1)	**2**	4	**4**

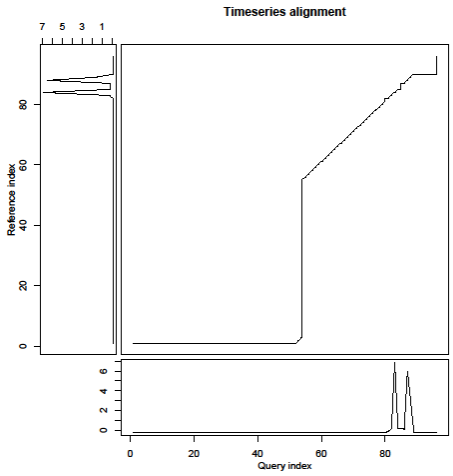
b) Blocked DTW matrix

Figure 34. Traditional DTW matrix

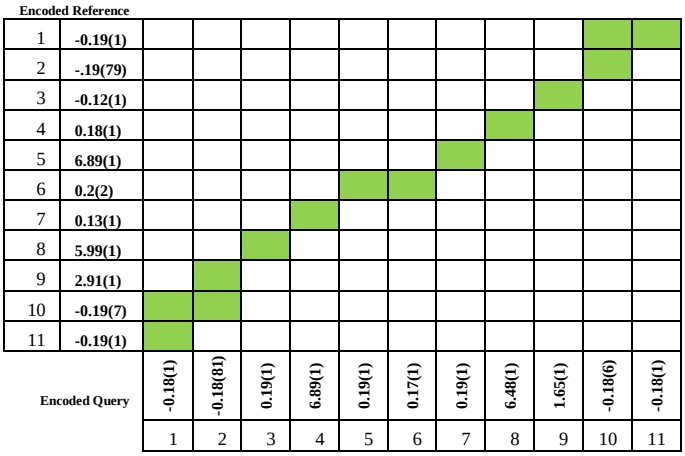
Figure 34, displays a regular DTW matrix and a blocked DTW matrix that one of the input time series is more-than-two-valued repetitive series. In this example the bottom-right value (6) in the block on the 3rd row (the blue block) is not equal to the corresponding cell value in BDTW matrix (7). The reason is that all the repetitive values in a block do not necessarily have the identical role in calculation of the values of the cells. By following the cells value calculation in the traditional matrix we notice that the second value in repetitive values (second 3 in the repetition) affects the calculation of the value 6 in the bottom right of the blue block. The effect of inside repetitive values obtains the optimal path with minimum DTW distance and ignoring this effect in BDTW might cause to different block distance. Since this distance is not calculated based on the optimal path inside the block, it might be higher or equal to the optimal path distance. Because of this fact, BDTW for more-than-two-valued repetitive time series is an upper bound of DTW distance and the algorithm 5.1 can be referred as BDTW_ Upper Bound (BDTW_UB) for this type of time series.

Figure 35, shows a graphical comparison of DTW and BDTW_UB on an example. Two time series are selected form first 2 rows of "ElecticDevices" dataset (from UCR time series archive). These time series have value repetition and their length is 96. Therefore, traditional DTW matrix for these time series is a 96 by 96 matrix (Figure 35a) and after performing the calculations based on traditional DTW algorithm, the exact DTW distance is 1.858176.

After the encoding these time series their length shrink to 11. Therefore, BDTW matrix will be a 11 by 11 matrix (Figure 35b) and after performing the calculations based on BDTW_UB algorithm, the BDTW distance is 1.858309. Note that as expected, this distance is higher than DTW but it is a very close approximation. For time series with high repetition rate, on one hand since BDTW shrinks the size of the cost matrix, it is faster than DTW (with less computational cost) and the distance calculated by this algorithm is a very close approximation of DTW distance. For time series without any repetitive values, BDTW algorithm obtains the same distance as traditional DTW.



a) DTW alignments



b) BDTW alignments

Figure 35. Graphical comparison of DTW and BDTW UB alignments

Furthermore, it is possible to change the formula in BDTW algorithm to obtain a lower bound of DTW distance. If we change the formula in the cases that the path is coming from diagonal to $\min(A,B) * (a-b)^2$, it will guaranty that the bottom-right value of the block is less than or equal to exact DTW distance of that block. Therefore, by this formulation the total BDTW distance of the matrix is lower bound of exact DTW distance. Algorithm 5.2, which is referred as BDTW_ Lower Bound (BDTW_LB), considers this change in the formula and obtains a DTW lower bound distance for time series with repetition of values.

Algorithm 5.2 Blocked_DTW_LB(series1, series2)

Required: series1 & series2 ←reduced time series in a 2xn data frame. The first row of the data frame is the values of the reduced time series, the second row is the number of repetitions.
Note: *A* and *B* represent the values of the points in the data reduced series, *a* and *b* are the number of times these values are repeated.
Ensure: Output the distance *d* between series1 and series2

1: $l_a \leftarrow$ length(series1), $l_b \leftarrow$ length(series2)
2: $D(1{:}l_a, 1{:}l_b) \leftarrow \infty$
3: $D_{1,1} \leftarrow (A_1\text{-}B_1)^2$
4: **for** i ← 1 to l_a
5: $D_{i,1} \leftarrow A_i(a_i\text{-}b_1)^2$
6: **end for**
7: **for** j ← 1 to l_b
8: $D_{1,j} \leftarrow B_j(a_1\text{-}b_j)^2$
9: **end for**
10: **for** i ← 2 to l_a
11: **for** j ← 2 to l_b
12: top ← $D_{i-1,j} + A_i(a_i\text{-}b_j)^2$
13: diagonal ← $D_{i-1,j-1} + (\min(A_i, B_j))(a_i\text{-}b_j)^2$
14: left ← $D_{i,j-1} + B_j(a_i\text{-}b_j)^2$
15: $D_{i,j} \leftarrow$ min(top, diagonal , left)
16: **end for**
17: **end for**
18: **return** $D_{la,lb}$

5.3.4 Constrained Blocked Dynamic Time Warping

As discussed in Section 2.4.1, one of the approaches to accelerate DTW is using Constrained DTW. In this method a warping windows is defined by the user to limit the cells calculation in the DTW cost matrix. Although Constrained DTW does not necessarily obtains the optimal path and exact DTW distance,

it is still widely used for time series classification and clustering. Because it is faster than DTW and usually obtains close approximation of the exact DTW with usually same or better level of accuracy in time series classification. Constrained warping window can be applied in BDTW. Algorithm 5.3 shows the constrained BDTW. In this algorithm, first the input time series are encoded time series. Assuming a_i and b_j timestamps as ta_i and tb_j, in the 12th line of the algorithm, $|ta_i - tb_j|$ is defined as the gap. In the 13th line of the algorithm if the gap is larger than a specific value (calculated based on defined warping window), the cell value would be set to infinity to avoid the warping path crossing the defined boundaries.

When a cell in the BDTW matrix is considered to be within the boundary, we are assuming that all the corresponding traditional DTW are also inside that boundary. This is not always a correct assumption. This fact causes a possible difference between Constrained DTW and Constrained BDTW results (even with same warping window).

Because of limiting the search area of warping path on reduced size BDTW matrix, Constrained BDTW calculation is both faster than Constrained DTW and also BDTW. It can be used as another approximation of DTW for time series with repetition of values.

Figure 36, presents the graphical comparison of DTW, DTW Constrained and BDTW Constrained alignments.

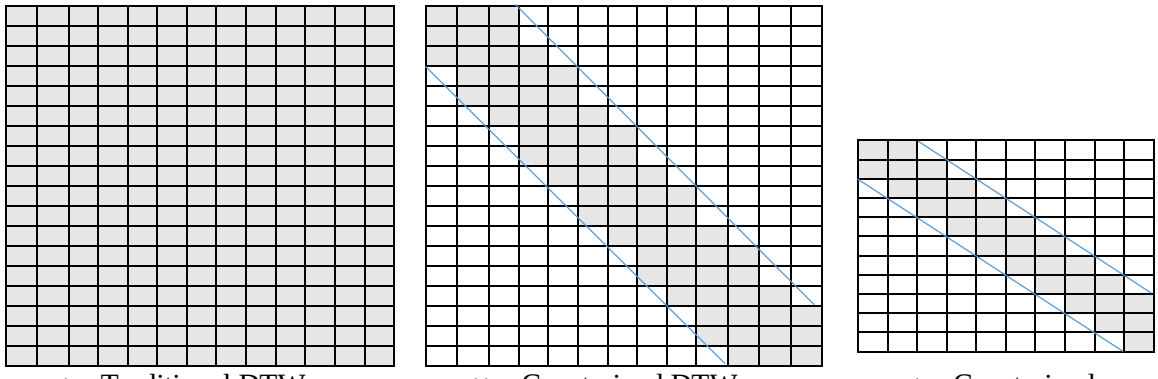


Figure 36. Graphical comparison of DTW, DTW Constrained and BDTW Constrained alignments

Note that Constrained BDTW (with any defined warping window) outputs an upper bound of DTW. The reason is that constrained BDTW obtains a warping path between two time series and any warping path between two time series is an upper bound of their optimal path (Constrained_BDTW>=DTW)

Algorithm 3 Constrainted_BDTW(series1, series2, w)

Required: series1 & series2 ←reduced time series in a 2xn data frame. The first row of the data frame is the values of the reduced time series, the second row is the number of repetitions.
Note: *A* and *B* represent the values of the points in the data reduced series, *a* and *b* are the number of times these values are repeated.
w ←warping windows
Ensure: Output the distance *d* between series1 and series2

```
 1: l_a ← length(series1), l_b ← length(series2)
 2: D(1:l_a, 1:l_b) ← ∞
 3: D_{1,1} ← (A_1-B_1)^2
 4: for i ← 1 to l_a
 5:     D_{i,1} ← A_i(a_i-b_1)^2
 6: end for
 7: for j ← 1 to l_b
 8:     D_{1,j} ← B_j(a_1-b_j)^2
 9: end for
10: for i ← 2 to l_a
11:     for j ← 2 to l_b
12:         gap ← |ta_i – tb_j|
13:         if gap > w && (tb_{j-1} – ta_i > w || ta_{i-1} – tb_j > w)
14:             D_{i,j} ← ∞
15:         else
16:             top ← D_{i-1,j} + A_i(a_i-b_j)^2
17:             diagonal ← D_{i-1,j-1} + (min(A_i, B_j))(a_i-b_j)^2
18:             left ← D_{i,j-1} + B_j(a_i-b_j)^2
19:             D_{i,j} ← min(top, diagonal , left)
20:         end ifelse
21:     end for
22: end for
23: return D_{la,lb}
```

5.3.5 Using BDTW for DTW Approximation of Any-Valued Time Series

In Section 5.3.3, BDTW is introduced as a close approximation of DTW which can significantly reduce DTW calculation for any-valued repetitive time series. But what about time

series without values repetition? In this section a new method is proposed to make it possible to use BDTW on time series with no values repetition. The idea is performing BDTW after applying a representation method on time series which produces constant segments of time series (i.e. PAA or APCA).

The constant segments in PAA or also APCA representation in fact represent the repetition a values and the length of the segment represents the number of value repetition. Therefore BDTW is applicable on top of constant segments representation methods.

In Section 5.4.3, it is shown that the APCA-BDTW approximation method beats the traditional approach of DTW approximation based on PAA-DTW-Projection. Note that by using BDTW after PAA or APCA, there is no need for projection step any more. Figure 38, shows the graphical comparison of approximation methods (PAA-DTW-Projection, PAA-BDTW, APCA-BDTW).

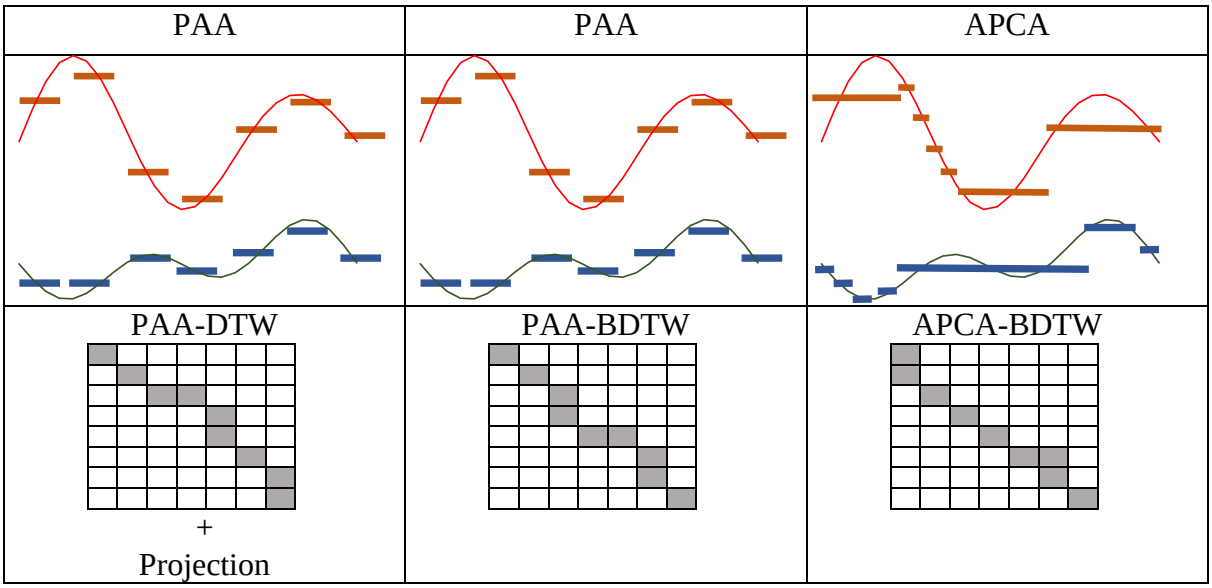


Figure 37. Graphical comparison of approximation methods (PAA-DTW-Projection/PAA-BDTW/APCA-BDTW)

Number of segments (the length of time series) after representation is controllable in PAA and APCA. It can be defined as the percentage of a time series length. The number of segments in PAA or APCA shows one minus the repetition rate in BDTW. For example if the original length of a time series is 200 and the number of segments is considered as 10%, then we would have 20 segments. After encoding these transformed time series, the encoded time series length will also be 20 and the repetition rate is 90% (1-10%).

The following formula can be used to measure the percentage of error for a distance approximation method:

$$Distance\ Error = \frac{|Approximate\ Distance - True\ DTW\ Distance|}{True\ DTW\ Distance} * 100 \qquad (5.4)$$

APCA–BDTW approximation performance (or PAA-BDTW performance) can be evaluated using this formula and compared with performance of other approximation techniques such as PAA-DTW-Projection.

5.3.6 Using BDTW as a Pruning Technique to Calculate Exact DTW

In addition to approximation methods in DTW calculation, pruning techniques are used to accelerate the calculation and obtaining the exact DTW distance. Using lower bounding and upper bounding for pruning unhopeful cells in the exact DTW calculation are two important methods in this category.

5.3.6.1 Using BDTW_LB as a New Lower Bound

In Section 2.4.2, Lower Bounding (LB) and important LB techniques (such as Kim, Yi and Keogh lower bounds) are presented. BDTW_LB can be considered as another lower bound technique. In a cases that a user prefers to have exact DTW distance (rather than an approximation) for time series with high level of repetition, BDTW_LB can be used to reduce the computation

time of the calculations. Lower bounds performances can be evaluated based on their own computation time plus their pruning power. BDTW_LB's computation complexity is quadratic on reduced time series length, however it is a very tight lower bound with high pruning power.

5.3.6.2 Using BDTW_UB as a Upper Bound in PruneDTW

Squared ED is the upper bound that PrunedDTW uses for pruning unhopeful alignments (cells) in the DTW matrix. BDTW_LB can be another alternative for an upper bound to be used in PrunedDTW algorithm. On one hand BDTW_LB has higher time complexity than ED, but on the other hand it has very tighter upper bound than ED. For the time series with high repetition rate, BDTW_LB pruning power will cover its high computation complexity.

Figure 37, displays an example of pruning unhopeful cells in DTW matrix with different upper bounds. Figure 37c, uses the exact DTW as the upper bound for pruning. The exact DTW distance is unknown before filling the DTW matrix, but in this case we assume that this value is imaginary given to us before starting the calculations in DTW matrix. Because of that sometimes the exact DTW distance as upper bound is called Oracle. Oracle upper bound obtains the highest possible pruning power in PrunedDTW. In this example BDTW_UB obtains much better pruning power than ED and almost same level of pruning as Oracle upper bound.

a) UB: ED b) UB:BDTW c) UB:exact DTW

Figure 38. An example of pruning unpromising alignments using different upper bounds (ED, BDTW and exact

5.4 Experiments and Results

In this section, the performance of BDTW (and its variants) is evaluated on two case studies and also 85 benchmark datasets in UCR Archive [43]. Five different application areas of BDTW are evaluated in this section: Firstly, using BDTW_UB as an approximation method of DTW for time series with high value repetition. Secondly, using a constraint version of BDTW. Thirdly, using BDTW_UB on top of APCA or PAA representation methods to approximate DTW distance for any time series. Fourthly, using BDTW_LB as a tight lower bound for pruning the time series that could not be best match in order to accelerate searches. Fifthly, using BDTW_UB for pruning unhopeful warping alignments to speed up the all pairwise DTW matrix calculation.

5.4.1 Case Study 1: Power Consumption Classification

To illustrate the effectiveness of BDTW over AWap and traditional DTW methods, we use a case study to determine types of power consumption. The Almanac of Minutely Power dataset Version 2 (AMPds2) is utilized as the data source for this study. AMPds2 is a dataset made available by [71] to researches to test models, systems and algorithm on real world data. Available datasets have been pre-cleaned to provide comparable accuracy and speed results amongst different researchers.

Over a two-year period, data points have been collected for a single house in Canada in AMPds2. These data points include information about electricity, water, and natural gas usage. For the propose of this study, only the data points related to electrical power consumption are used. Since this dataset contains information for a two-year period, each component can be broken into 730 time series, one per day. This process is followed for the components of interest (B1E, B2E, BME, CDE, CWE, DNE, DWE, EBE, EQE, FGE, FRE, HPE, HTE, OFE, OUE, TVE, UTE,

WOE). These components correspond to different sources of electrical consumption such as dining room, furnace, outside, etc. Figure 39, shows the example of some of these time series. Note that since in some periods of time some components are not active, their power consumption are zero. Therefore time series commonly have repetition of zero and non-zero values and it makes sense to apply AWarp and BDTW on them.

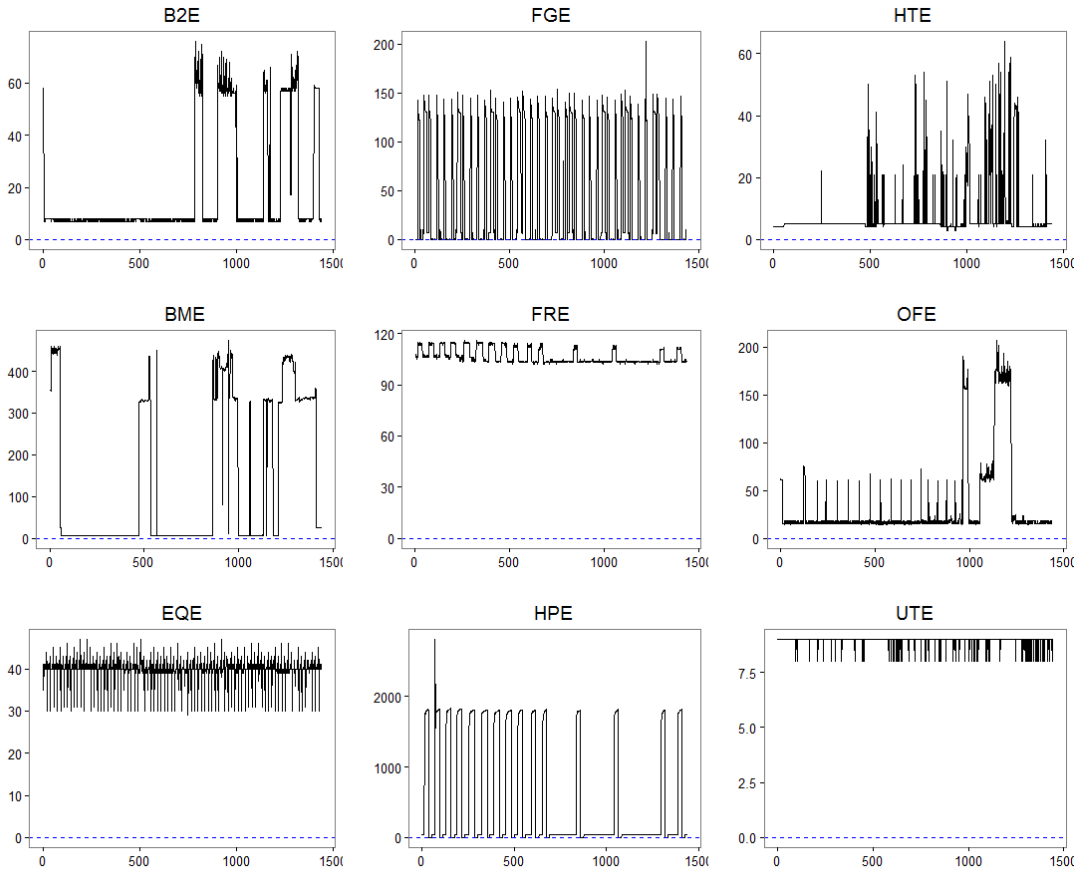


Figure 39. The example of some of electricity components time series

Based on this information, the goal is to be able to determine the source of a series. For demonstration purposes, we limit our testing to 30 days' worth of information for each component. The data is tested using Leave Out One Cross-Validation. The classification problem is run using BDTW, AWarp, DTW, Euclidian Distance, Constrained DTW, and Constrained BDTW. The result is presented in Table V.

TABLE V. THE COMPARISON OF ACCURACY AND PROCESSING TIME- POWER CONSUMPTION

Method	**ED Time**	**DTW**	**AWarp**	**BDTW**	**CDTW**	**CBDTW**
Accuracy	0.561	0.807	0.806	0.807	0.761	0.776
Processing time in Seconds	51.959	8514.197	3237.528	877.575	936.562	307.496

The result shows that ED accuracy (0.561) is much lower than BDTW accuracy (0.807). BDTW accuracy (0.807) is exactly same as DTW accuracy (0.807) and slightly better than AWarp (0.806). BDTW is roughly 10 times faster than DTW and 3.7 times faster than AWarp. The comparison of CDTW and CBDTW with the same percentage of warping window (10%), shows that CBDTW is almost 3 times faster than CDTW with slightly better accuracy.

5.4.2 Case Study 2: Household Electrical Measurements

To further show the performance and application of the BDTW method, we use data collected on household electrical measurements form [72]. This dataset contains information about the electrical usage of different appliances within 21 houses. For the purposes of this case study, we focus on two households, houses 3 and 8. To get extract the information for each household from the original data we use the supplied UNIX time to split the data into time that represent a full day. The data is collected on average each day consists of 10,800 data points, which represents one reading every eight seconds. For this study, we limited the data to the first 100 full days for each household and just the data that occurred between 5:00pm and 11:00pm (consumption peak hours).

Using this subset of data, we aim to demonstrate the capabilities of BDTW for classifying a household in two different contexts, aggregate and freezer power consumption. Figure 40, shows two random time series of aggregate and freezer power consumption. This figure shows that for

aggregate time series we can see the repetition of values but all repetition values are non-zero. On the other hand in Freezer time series we see the repetition of zero and non-zero values.

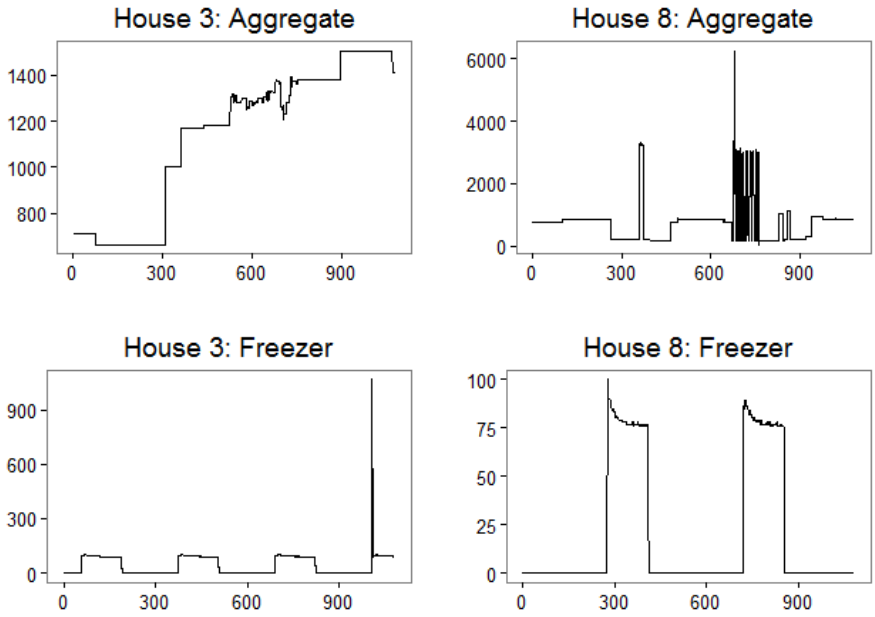


Figure 40. Two random time series of aggregate and freezer power consumption

The classification problem is run using BDTW, AWarp, DTW, Euclidian Distance, Constrained DTW, and Constrained BDTW. The data is tested using Leave Out One Cross-Validation. The result is presented in Table VI. In both cases (Aggregate and Freezer), the accuracy of DTW, AWarp and BDTW are equal and higher than accuracy of ED. Regarding the processing time for Aggregate case, since there is no repetition of zero values in time series AWarp is not useful. AWarp processing time (174.98s) is even higher than DTW processing time (143.85s). BDTW processing time is 57.41s, which means BDTW is 3 times faster than AWarp and 2.5 times faster than DTW. In Freezer case, since there is repetition of zero value in the time series (along with repetition of non-zero values), AWarp is to some extend useful in reduction of processing time. However since BDTW utilizes the repetition of any values (zero and non-zero values), it is

significantly faster than AWarp. BDTW is 48.5 times faster than DTW and 4 times faster than AWarp. The efficiency improvement of BDTW in Freezer case is more than Aggregate case, because the repetition rate of Freezer dataset (91.13%) is more than Aggregate repetition rate (49.87%). In both cases CBDTW also obtains similar or better accuracy than CDTW with less processing time. CBDTW is more than 2 times faster than CDTW in Aggregate case and 46.6 time faster than CDTW in Freezer case.

TABLE VI. THE COMPARISON OF ACCURACY AND PROCESSING TIME-HOUSEHOLD ELECTRICAL MEASUREMENTS

Dataset	**rr**	**Method**	**ED**	**DTW**	**AWarp**	**BDTW**	**CDTW**	**CBDTW**
HOUSE3,8 Aggregate	49.87%	Accuracy	0.87	0.90	0.90	0.90	0.89	0.87
		Processing time	5.67	143.85	174.98	57.41	71.93	33.20
HOUSE3,8 Freezer	91.13%	Accuracy	0.64	0.86	0.86	0.86	0.73	0.77
		Processing time	17.62	1394.8	117.23	28.76	697.42	14.95

5.4.3 BDTW_UB as an Approximation Method of DTW

Figure 41, shows the classification process time of DTW over BDTW based on different repetition rates (the x axes). Each point in this figure represents a dataset from UCR Archive [43] and this figure shows that the superiority of BDTW speed over DTW speed increases significantly when repetition rate decreases. When a dataset's repetition rate is greater than 0.25 we can expect gain in process time. There are 14 datasets (listed in Table VII) with repetition rate of greater than 0.25 that in all of them BDTW is faster than traditional DTW. The highest repetition rate is for "SmallKitchen Appliances" which BDTW is approximately 3.5 times faster than DTW.

TABLE VII. THE LIST OF DATASETS WITH SHORTER PROCESSING TIME USING BDTW

Dataset name	Repetition rate	Speed up percentage
SmallKitchenAppliances	86.4%	3529%
LargeKitchenAppliances	81.6%	1914%
ScreenType	77.5%	1421%
Computers	70.9%	844%
Earthquakes	68.1%	747%
RefrigerationDevices	63.9%	505%
ElectricDevices	57.6%	292%
wafer	48.8%	206%
FaceFour	47.2%	199%
Two_Patterns	34.4%	151%
uWaveGestureLibrary_Y	31.8%	148%
uWaveGestureLibraryAll	28.2%	120%
uWaveGestureLibrary_X	26.2%	139%
uWaveGestureLibrary_Z	25.6%	132%

Using a nonlinear exponential regression curve fitting method, the following equation can be used to estimate the processing time gain based on the repetition rate:

$$Speed\ Gain = e^{4.6348*(repetion\ rate)^2}$$

In Figure 42, the comparison BDTW and DTW in terms of classification accuracy is presented. This figure shows that BDTW is very close approximation of DTW. In many cases (63 out of 85) both methods result in same classification error. In some case (9 out of 85) BDTW classification error is slightly less than DTW and in 13 out of 85 datasets DTW has slightly less error.

Using BDTW for time series with high repetition level (low repetition rate), will have almost same accuracy but shorter process time than traditional DTW.

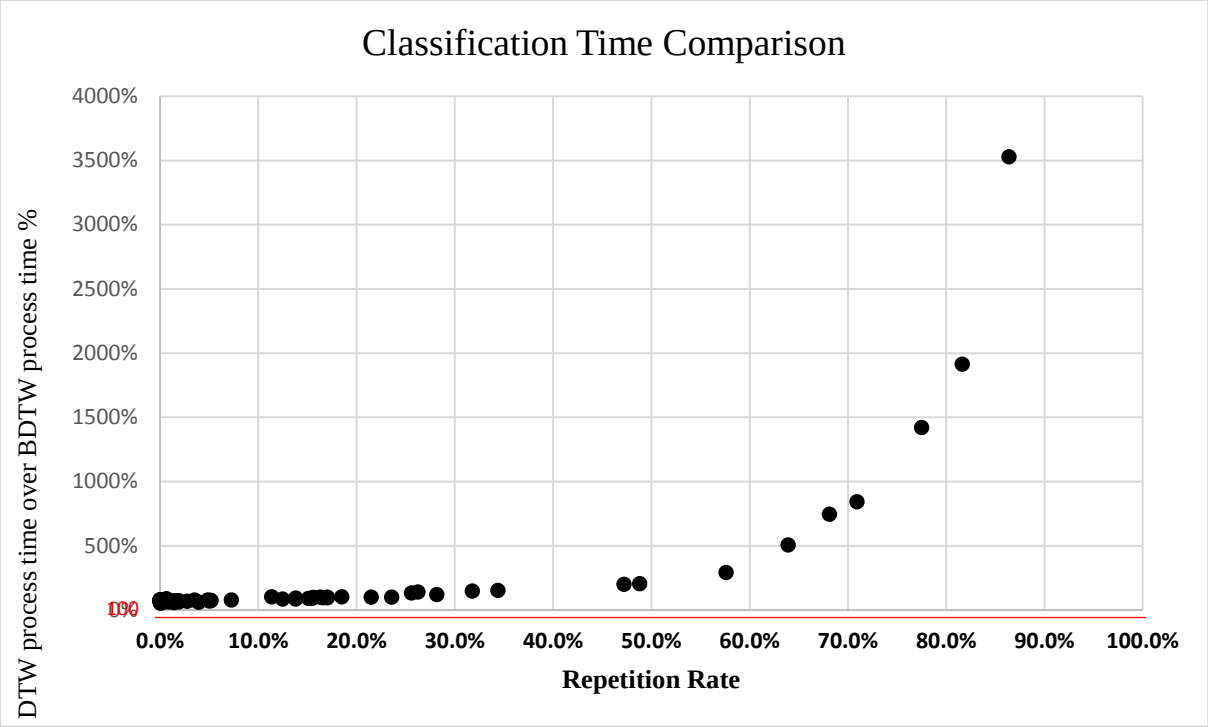


Figure 41. Speed gain of BDTW for 85 time series datasets in UCR archive

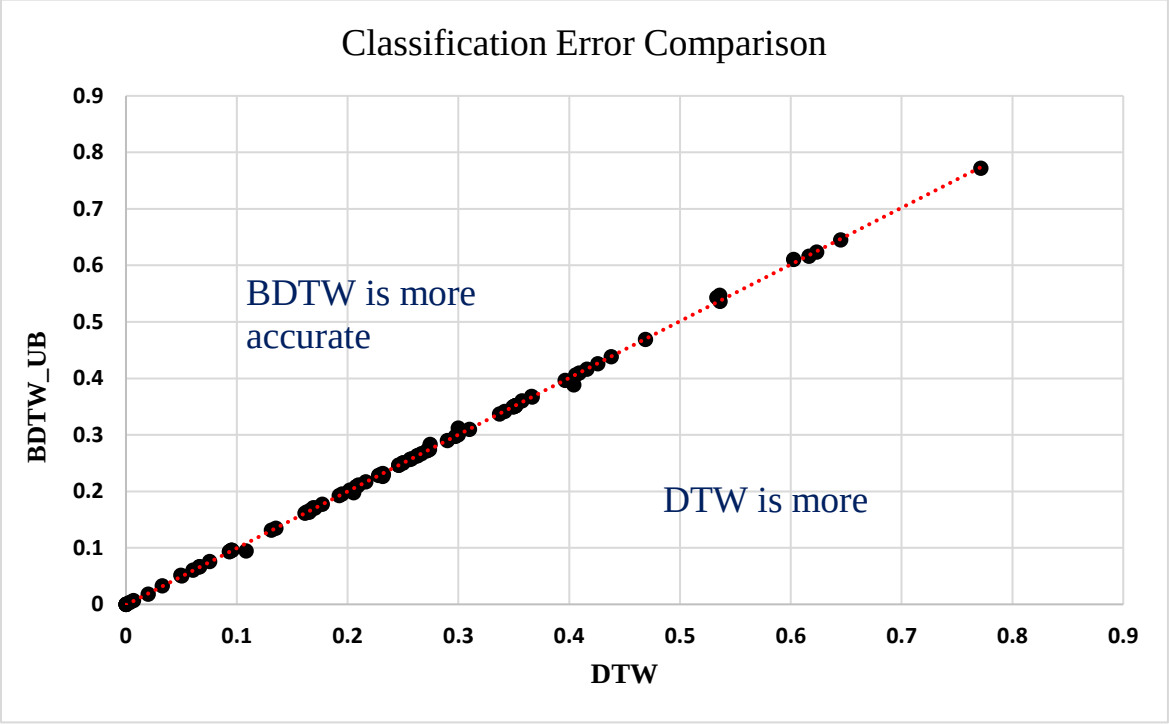


Figure 42. The comparison BDTW and DTW in terms of classification error for 85 time series datasets in UCR

5.4.4 CBDTW_LB: Constrained Block DTW Lower Bound

In Table VIII, the classification accuracy of Constrained BDTW (CBDTW) is compared with classification accuracy of DTW, Constrained DTW (CDTW) and BDTW for 14 time series with high repetition rate in UCR Archive. This table shows that similar to BDTW, CBDTW is a close approximation of DTW and in many cases it performs same or better than BDTW (12 out 14 datasets).

In Table IX, the process time of DTW, Constrained DTW (with warping window of 10% of the length of time series), BDTW and Constrained BDTW (with warping window of 10% of the length of time series) for datasets with high repetition rate are compared. On average, Constrained BDTW is twice faster than BDTW.

TABLE VIII. COMPARISON ACCURACY OF DTW, BDTW, CONSTRAINED DTW AND CONSTRAINED BDTW

Dataset	**Repetition Rate**	**Accuracy**			
		DTW	**CDTW**	**BDTW**	**CBDTW**
SmallKitchenAppliances	86.40%	64%	61%	64%	73%
LargeKitchenAppliances	81.60%	79%	71%	80%	72%
ScreenType	77.50%	40%	43%	39%	39%
Computers	70.90%	70%	64%	69%	65%
Earthquakes	68.10%	74%	74%	74%	74%
RefrigerationDevices	63.90%	46%	46%	46%	51%
ElectricDevices	57.60%	60%	62%	61%	64%
wafer	48.80%	98%	98%	98%	100%
FaceFour	47.20%	83%	83%	83%	88%
Two-Patterns	34.40%	100%	100%	100%	100%
uWaveGestureLibrary-Y	31.80%	63%	66%	63%	70%
uWaveGestureLibraryAll	28.20%	89%	89%	91%	95%
uWaveGestureLibrary-X	26.20%	73%	74%	73%	80%
uWaveGestureLibrary-Z	25.60%	66%	67%	66%	70%

TABLE IX. COMPARISON PROCESSING TIME OF DTW, BDTW, CONSTRAINED DTW AND CONSTRAINED BDTW

Dataset	rr	Processing time(s)				CBDTW Speed-up ratio		
		DTW	CDTW	BDTW	CBDTW	DTW	CDTW	BDTW
SmallKitchenAppliances	86.40%	715.4	770.7	20.3	16	44.7	48.2	1.3
LargeKitchenAppliances	81.60%	718.9	571.4	37.6	22.8	31.5	25.1	1.6
ScreenType	77.50%	647.7	514.8	45.6	27.7	23.4	18.6	1.6
Computers	70.90%	312.9	297.5	37.1	24.7	12.6	12	1.5
Earthquakes	68.10%	124.7	118.5	16.7	11.1	11.2	10.6	1.5
RefrigerationDevices	63.90%	758.7	1,016.3	150.2	86	8.8	11.8	1.7
ElectricDevices	57.60%	6,172.2	4,568.5	2,111	1,087	5.7	4.2	1.9
wafer	48.80%	1,120	375.4	543.5	278.5	4	1.3	2
FaceFour	47.20%	2.7	3	1.4	0.5	5.3	5.9	2.7
Two-Patterns	34.40%	724.8	317.4	478.5	195.1	3.7	1.6	2.5
uWaveGestureLibrary-Y	31.80%	2,496	967.2	1,686	740.1	3.4	1.3	2.3
uWaveGestureLibraryAll	28.20%	26,655	28,464	22,290	13,844	1.9	2.1	1.6
uWaveGestureLibrary-X	26.20%	2,688	1,064.9	1,932	850.1	3.2	1.3	2.3
uWaveGestureLibrary-Z	25.60%	2,335	925.1	1,774	780.6	3	1.2	2.3

5.4.5 BDTW_UB and APCA as an Approximation Method for Any-Valued Time Series

From each datasets in UCR Archive 20% of all-pairwise time series are randomly selected and for each time series pair, true DTW distance and approximation distances are calculated. To make the process time linear, for calculating DTW approximation we use $\sqrt{n}$ as the number of segments in time series representation (where n is the length of time series). Using formula 5.4, we calculate the error between exact DTW distance and approximation methods. The approximation methods that we compare are Abstraction (using PAA-DTW-Projection), PAA-BDTW_UB and APCA-BDTW_UB. The result is presented in Figure 43. The x axes is the name of the datasets and the y axes is the mean of error. This figure shows that in most datasets PAA-BDTW_UB and APCA-BDTW_UB outperform PAA-DTW based approximation method (i.e., Abstraction).

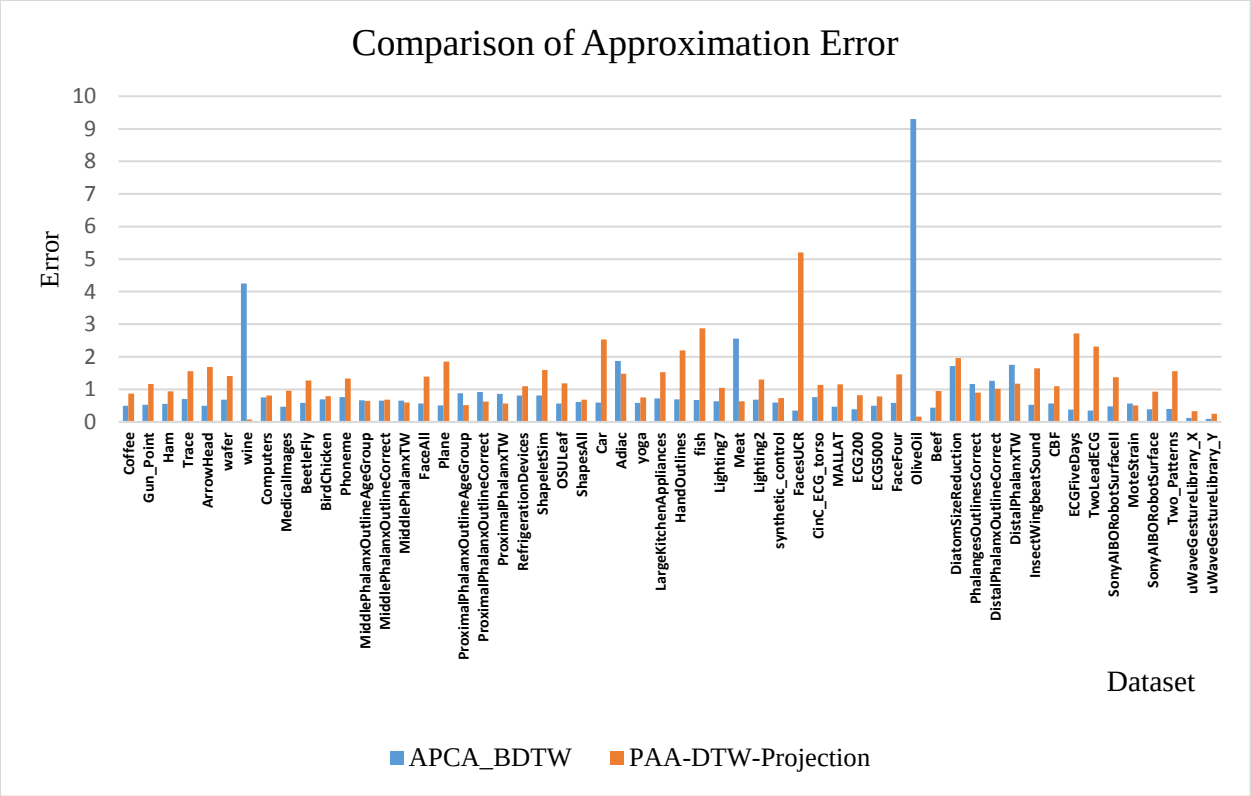


Figure 43. Comparison of approximation error between APCA BDTW and PAA-DTW-Projection

5.4.6 BDTW_LB: a New Lower Bound to Increase the Efficiency of DTW

BDTW_LB can be used as lower bound to accelerate the similarity search when we want to have the exact DTW (not approximation). In Table X, the power of BDTW_LB in terms of speed up the process time of DTW is compared with Kim_LB and Yi_LB lower bounds on 6 highest repetitive datasets in the UCR Archive. When speed up is less than 1, it means using lower bound does not help in reducing the classification process time.

BDTW_LB performs better than both Kim_LB and Yi_LB in all of these 6 datasets with high repetition rate. Usually, as repetition rate of a dataset increases, BDTW_LB performance also increases. Note that Keogh lower bound is not compared here, because it only works on

constrained DTW but here we evaluate lower bound performance without warping windows constrain.

TABLE X. COMPARISON THE EFFICIENCY OF BDTW LOWER BOUND WITH KIM AND YI LOWER BOUNDS

Dataset	Repetition Rate	Speed-up ratio		
		Kim_LB	**Yi_LB**	**BDTW_LB**
SmallKitchenAppliances	86%	0.89	1.25	16.87
LargeKitchenAppliances	82%	0.99	1.36	10.84
ScreenType	78%	0.95	2.13	5.24
Computers	71%	0.88	1.65	4.59
Earthquakes	68%	0.67	0.59	2.32
RefrigerationDevices	64%	0.88	1.03	2.25

5.4.7 Using BDTW_UB to Speed up All-pairwise DTW Matrix Calculation

PrunedDTW code is used to measure the performance of BDTW_UB as the applied upper bound for pruning unpromising alignments in all-pairwise DTW matrix calculation. In Figure 44, the processing time of DTW without pruning, DTW pruned by ED as the upper bound, DTW pruned by BDTW_UB as the upper bound and OracleDTW are compared on 6 datasets with high repetition rate. The OracleDTW is DTW pruned by the exact DTW distance as the upper bound. Note that even in practice we do not have exact DTW distance to use it as the upper bound for pruning, here we assume that it is imaginary given to us just to evaluate the highest possible pruning power. Therefore, OracleDTW presents the best possible performance of PrunedDTW using the optimal upper bound.

This figure shows that for the high repetition rate problems, BDTW_UB outperformed DTW and also DTW pruned by ED. The higher repletion rate in a problem the better perforce of BDTW_UB. When repetition rate increases to more than 0.8, BDTW_UB's performance gets very close to OracleDTW's performance (the best possible performance of PruedDTW).

The process time of BDTW_UB is summation of two times: the time for calculating BDTW_UB distance and the time for pruning and calculating DTW matrix. Since BDTW_UB is a close approximation of exact DTW, its required time for pruning and matrix calculation is very close to OracleDTW. However the required time for calculating BDTW_UB distance is the overhead that adds to process time. When repetition rate is high enough this overhead will be justified in total process time because BDTW_UB distance calculation will be fast.

Note that in some problems (for example ScreenType, Earthquakes and RefrigerationDevices) using ED as the upper bound for pruning does not help in reducing the process time. Because for these problems although ED is fast to calculate but, it does not provide a powerful pruning level.

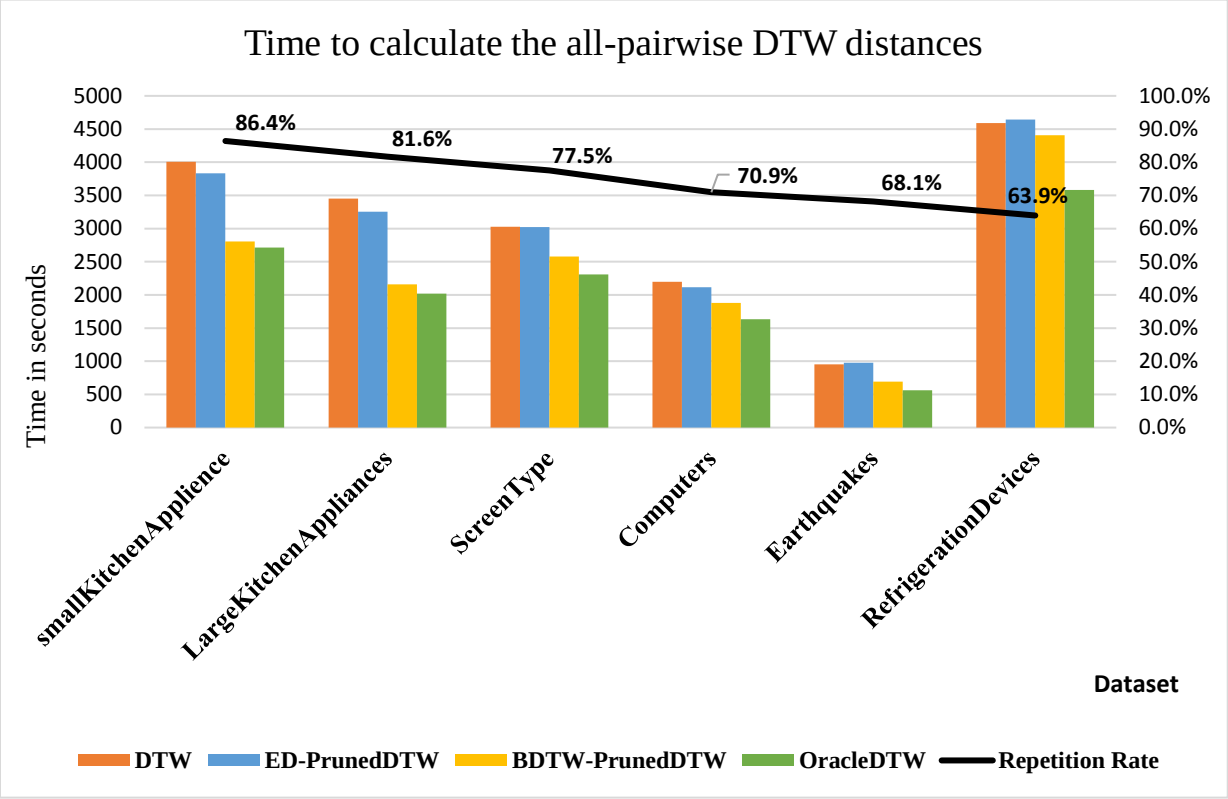


Figure 44. Comparison processing time to calculate the all-pairwise DTW distances using DTW without pruning.

6 CONCLUSION AND FUTURE WORKS

6.1 Summary and Results

This dissertation focuses on the time series classification and tries to speed up Dynamic Time Warping calculations by introducing a new time series representation (transformation) and two time series similarity measures which work on transformed time series. In this context, the contributions of this dissertation are as follows:

Control Chart Approximation (CCA): CCA representation approximates and reduces the length of raw time series by creating a vector of segments with single values and waiting times. The output of CCA representation is the input of 1-NN 3D DTW classifier.

3 Dimensional Dynamic Time Warping (3D DTW): 3D DTW is the version of traditional DTW in a 3 dimensional space. It is used to measure the distance between two transformed time series based on CCA representations.

Using 85 time series, benchmark datasets from UCR archive, including 28 long time series datasets in an exhaustive evaluation, it is shown that 1-NN 3D DTW is orders of magnitude faster than the state-of-the-art implementation of 1-NN DTW. It has better or similar level of classification accuracy for long time series in the experiment.

The trade-off between classification accuracy against processor time and computational cost, 1-NN 3D DTW (in comparison to the state-of-the-art classifiers) is competitive and performs reasonably well.

Almost all of 1-NN DTW extension methods for accuracy or efficiency improvements such as global warping constraints and weighting techniques, are also applicable on 1-NN 3D DTW.

Blocked Dynamic Time Warping (BDTW): BDTW is a new similarity measure which works on run-length encoded time series representation. BDTW takes advantage of values repetition in time series to shrink the size of DTW matrix for a reduction of processing time. BDTW algorithms offer an upper bound and a lower bound of DTW. BDTW upper bound is a close approximation of exact DTW and significantly reduces the processing time of calculations on time series with high levels of values repetition.

The Combination of BDTW upper bound and APCA provides a close approximation of DTW distance, even for time series with low levels of values repetition or without any value repetition. APCA-BDTW outperforms the approximation method based on PAA-DTW Projection.

BDTW can also be used as a pruning technique to accelerate calculation of the exact DTW. BDTW lower bound can be used for pruning of unpromising candidates in similarity search and it performs better Kim and Yi lower bounds for time series with high levels of values repetition. BDTW upper bound can also be used for pruning of unpromising alignments in all-pairwise DTW calculation and performs better than ED (as the UB) for time series with high levels of values repetition.

In Constrained BDTW Algorithm, a warping constraint is used to limit the search area in BDTW matrix and for time series with high repetition rates, Constrained BDTW is faster than Constrained DTW with similar or better accuracy. BDTW is also extendable to multidimensional time series warping.

6.2 Future Works

Future improvements in speed and accuracy of time series analysis might be accomplished by the following proposed tasks:

Learning the optimal parameter: Developing tuning methods for parameter *s* in CCA and 1-NN 3D DTW, according to the target time series, could further improve the efficiency of this technique on some datasets.

Multidimensional time series: Both 3D DTW and BDTW are extendable to multidimensional time series warping. Investigating the specification of multidimensional time series warping on these methods can be considered as the future work of this study.

Time series analytics tasks: The focus of this study is time series classification (TSC). However, the application of the proposed similarity measures (3D DTW and BDTW) is not limited to TSC. These methods can also straightforwardly be used in time series clustering, motif discovery, or discord discovery (anomaly detection).

CITED LITERATURE

[1] Faloutsos, Christos, Mudumbai Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. Vol. 23, no. 2. ACM, 1994.

[2] Sakoe, Hiroaki, and Seibi Chiba. "Dynamic programming algorithm optimization for spoken word recognition." Acoustics, Speech and Signal Processing, IEEE Transactions on 26, no. 1 (1978): 43-49.

[3] Berndt, Donald J., and James Clifford. "Using Dynamic Time Warping to Find Patterns in Time Series." In KDD workshop, vol. 10, no. 16, pp. 359-370. 1994.

[4] Keogh, Eamonn, and Shruti Kasetty. "On the need for time series data mining benchmarks: a survey and empirical demonstration." Data Mining and knowledge discovery 7, no. 4 (2003): 349-371.

[5] Chadwick, Neisha A., David A. McMeekin, and Tele Tan. "Classifying eye and head movement artifacts in EEG Signals." In 5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011), pp. 285-291. IEEE, 2011.

[6] Ding, Hui, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. "Querying and mining of time series data: experimental comparison of representations and distance measures." Proceedings of the VLDB Endowment 1, no. 2 (2008): 1542-1552.

[7] Lines, Jason, and Anthony Bagnall. "Time series classification with ensembles of elastic distance measures." Data Mining and Knowledge Discovery 29, no. 3 (2015): 565-592.

[8] Schmill, Matthew D., Tim Oates, and Paul R. Cohen. "Learned models for continuous planning." In AISTATS. 1999.

[9] Rakthanmanon, Thanawin, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. "Searching and mining trillions of time series subsequences under dynamic time warping." In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 262-270. ACM, 2012.

[10] Batista, Gustavo EAPA, Xiaoyue Wang, and Eamonn J. Keogh. "A Complexity-Invariant Distance Measure for Time Series." In SDM, vol. 11, pp. 699-710. 2011.

[11] Batista, Gustavo EAPA, Eamonn J. Keogh, Oben Moses Tataw, and Vinícius MA de Souza. "CID: an efficient complexity-invariant distance for time series. "Data Mining and Knowledge Discovery 28, no. 3 (2014): 634-669.

[12] Megalooikonomou, Vasileios, Qiang Wang, Guo Li, and Christos Faloutsos. "A multiresolution symbolic representation of time series." In Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on, pp. 668-679. IEEE, 2005.

[13] Ratanamahatana, Chotirat Ann, and Eamonn Keogh. "Three myths about dynamic time warping data mining." In Proceedings of SIAM International Conference on Data Mining (SDM'05), pp. 506-510. 2005.

[14] Keogh, Eamonn J., and Michael J. Pazzani. "Derivative Dynamic Time Warping." In SDM, vol. 1, pp. 5-7. 2001.

[15] Kulbacki, Marek, and Artur Bak. "Unsupervised learning motion models using dynamic time warping." In Intelligent Information Systems 2002, pp. 217-226. Physica-Verlag HD, 2002.

[16] Jeong, Young-Seon, Myong K. Jeong, and Olufemi A. Omitaomu. "Weighted dynamic time warping for time series classification." Pattern Recognition 44, no. 9 (2011): 2231-2240.
[17] Montgomery, Douglas C. Statistical quality control. Vol. 7. New York: Wiley, 2009.
[18] Itakura, Fumitada. "Minimum prediction residual principle applied to speech recognition." IEEE Transactions on Acoustics, Speech, and Signal Processing 23, no. 1 (1975): 67-72.
[19] Ratanamahatana, Chotirat Ann, and Eamonn Keogh. "Making time-series classification more accurate using learned constraints." SDM, 2004.
[20] Esling, Philippe, and Carlos Agon. "Time-series data mining." ACM Computing Surveys (CSUR) 45, no. 1 (2012): 12.
[21] Keogh, Eamonn, Li Wei, Xiaopeng Xi, Michail Vlachos, Sang-Hee Lee, and Pavlos Protopapas. "Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures. "The VLDB Journal—The International Journal on Very Large Data Bases 18, no. 3 (2009): 611-630.
[22] Kim, Sang-Wook, Sanghyun Park, and Wesley W. Chu. "An index-based approach for similarity search supporting time warping in large sequence databases." In Data Engineering, 2001. Proceedings. 17th International Conference on, pp. 607-614. IEEE, 2001.
[23] Yi, Byoung-Kee, H. V. Jagadish, and Christos Faloutsos. "Efficient retrieval of similar time sequences under time warping." In Data Engineering, 1998. Proceedings, 14th International Conference on, pp. 201-208. IEEE, 1998.
[24] Fu, Ada Wai-Chee, Eamonn Keogh, Leo Yung Lau, Chotirat Ann Ratanamahatana, and Raymond Chi-Wing Wong. "Scaling and time warping in time series querying." The VLDB Journal—The International Journal on Very Large Data Bases 17, no. 4 (2008): 899-921.
[25] Vail, Douglas, and Manuela Veloso. "Learning from accelerometer data on a legged robot." In Proceedings of the 5th IFAC/EURON symposium on intelligent autonomous vehicles. 2004.
[26] Junkui, Li, and Wang Yuanzhen. "Early abandon to accelerate exact dynamic time warping." Int. Arab J. Inf. Technol. 6, no. 2 (2009): 144-152.
[27] Agrawal, Rakesh, Christos Faloutsos, and Arun Swami. "Efficient similarity search in sequence databases." In International Conference on Foundations of Data Organization and Algorithms, pp. 69-84. Springer Berlin Heidelberg, 1993.
[28] Korn, Flip, Hosagrahar V. Jagadish, and Christos Faloutsos. "Efficiently supporting ad hoc queries in large datasets of time sequences." ACM SIGMOD Record 26, no. 2 (1997): 289-300.
[29] Chan, Kin-Pong, and AW-C. Fu. "Efficient time series matching by wavelets." In Data Engineering, 1999. Proceedings, 15th International Conference on, pp. 126-133. IEEE, 1999.
[30] Keogh, Eamonn, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. "Dimensionality reduction for fast similarity search in large time series databases." Knowledge and information Systems 3, no. 3 (2001): 263-286.
[31] Keogh, Eamonn, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. "Locally adaptive dimensionality reduction for indexing large time series databases." ACM SIGMOD Record 30, no. 2 (2001): 151-162.

[32] Cai, Yuhan, and Raymond Ng. "Indexing spatio-temporal trajectories with Chebyshev polynomials." In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp. 599-610. ACM, 2004.
[33] Fu, Tak-chung. "A review on time series data mining." Engineering Applications of Artificial Intelligence 24, no. 1 (2011): 164-181.
[34] André-Jönsson, Henrik, and Dushan Z. Badal. "Using signature files for querying time-series data." In Principles of Data Mining and Knowledge Discovery, pp. 211-220. Springer Berlin Heidelberg, 1997.
[35] Huang, Yun-Wu, and Philip S. Yu. "Adaptive query processing for time-series data." In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 282-286. ACM, 1999.
[36] Bagnall, Anthony J., and Gareth J. Janacek. "Clustering time series from ARMA models with clipped data." In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 49-58. ACM, 2004.
[37] Mörchen, Fabian, and Alfred Ultsch. "Optimizing time series discretization for knowledge discovery." In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pp. 660-665. ACM, 2005.
[38] Lin, Jessica, Eamonn Keogh, Stefano Lonardi, and Bill 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, pp. 2-11. ACM, 2003.
[39] Lin, Jessica, Eamonn Keogh, Li Wei, and Stefano Lonardi. "Experiencing SAX: a novel symbolic representation of time series." Data Mining and Knowledge Discovery 15, no. 2 (2007): 107-144.
[40] Lkhagva, Battuguldur, Yu Suzuki, and Kyoji Kawagoe. "Extended SAX: Extension of symbolic aggregate approximation for financial time series data representation." In Proceedings of Data Engineering Workshop, pp. 4A-i8. 2006.
[41] Bennett, Kristin P., Usama Fayyad, and Dan Geiger. "Density-based indexing for approximate nearest-neighbor queries." In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 233-243. ACM, 1999.
[42] Nath, Happy, and Ujwala Baruah. "Evaluation of Lower Bounding Methods of Dynamic Time Warping (DTW)." International Journal of Computer Applications 94, no. 20 (2014).
[43] Keogh, Eamonn, Xiaopeng Xi, Li Wei, and Chotirat Ann Ratanamahatana. "The UCR time series classification/clustering homepage." URL= http://www. cs. ucr. edu/~ eamonn/time_series_data (2006).
[44] Hamooni, Hossein, and Abdullah Mueen. "Dual-domain hierarchical classification of phonetic time series." In 2014 IEEE International Conference on Data Mining, pp. 160-169. IEEE, 2014.
[45] Schäfer, Patrick. "Scalable time series classification." Data Mining and Knowledge Discovery (2015): 1-26
[46] Bagnall, Anthony, Jason Lines, Jon Hills, and Aaron Bostrom. "Time-series classification with COTE: the collective of transformation-based ensembles."IEEE Transactions on Knowledge and Data Engineering 27, no. 9 (2015): 2522-2535.
[47] Schäfer, Patrick. "The BOSS is concerned with time series classification in the presence of noise." Data Mining and Knowledge Discovery 29, no. 6 (2015): 1505-1530.

[48] Schäfer, Patrick. "Towards time series classification without human preprocessing." In International Workshop on Machine Learning and Data Mining in Pattern Recognition, pp. 228-242. Springer International Publishing, 2014.
[49] Baydogan, Mustafa Gokce, George Runger, and Eugene Tuv. "A bag-of-features framework to classify time series." IEEE transactions on pattern analysis and machine intelligence 35, no. 11 (2013): 2796-2802.
[50] Rakthanmanon, Thanawin, and Eamonn Keogh. "Fast shapelets: A scalable algorithm for discovering time series shapelets." In Proceedings of the 13th SIAM international conference on data mining, pp. 668-676. 2013.
[51] Mueen, Abdullah, Eamonn Keogh, and Neal 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, pp. 1154-1162. ACM, 2011.
[52] Lin, Jessica, Rohan Khade, and Yuan Li. "Rotation-invariant similarity in time series using bag-of-patterns representation." Journal of Intelligent Information Systems 39, no. 2 (2012): 287-315.
[53] Senin, Pavel, and Sergey Malinchik. "Sax-vsm: Interpretable time series classification using sax and vector space model." In 2013 IEEE 13th International Conference on Data Mining, pp. 1175-1180. IEEE, 2013.
[54] Bagnall, Anthony, Aaron Bostrom, James Large, and Jason Lines. "The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version." arXiv preprint arXiv:1602.01711 (2016).
[55] Grabocka, Josif, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. In Proc. 20th SIGKDD, 2014.
[56] Marteau, Pierre-Francois. Time warp edit distance with stiffness adjustment for time series matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(2):306–318, 2009.
[57] Stefan, Alexandra, Vassilis Athitsos, and Gautam Das. The Move-Split-Merge metric for time series. IEEE Transactions on Knowledge and Data Engineering, 25(6):1425–1438, 2013.
[58] Górecki, Tomasz and Maciej Luczak. Using derivatives in time series classification. Data Mining and Knowledge Discovery, 26(2):310–331, 2013.
[59] Górecki, Tomasz and Maciej Luczak. Non-isometric transforms in time series classification using DTW. Knowledge-Based Systems, 61:98–108, 2014
[60] Kate, Rohit. Using dynamic time warping distances as features for improved time series classification. Data Mining and Knowledge Discovery, online first, 2015.
[61] Rakthanmanon, Thanawin and Eamonn Keogh. Fast-shapelets: A fast algorithm for discovering robust time series shapelets. In Proc. 13th SDM, 2013.
[62] Deng, Houtao, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. Information Sciences, 239:142–153, 2013.
[63] Baydogan, Mustafa Gokce, and George Runger. Time series representation and similarity based on local autopatterns. Data Mining and Knowledge Discovery, 30(2):476–509, 2016.
[64] Bagnall, Anthony, and Jason Lines. Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery, 29:565–592, 2015.

[65] A. Mueen, N. Chavoshi, N. Abu-El-Rub, H. Hamooni, and A. Minnich, "Awarp: Fast Warping Distance for Sparse Time Series." In Data Mining (ICDM) IEEE, 2016, PP. 350-350.
[66] D. F. Silva and G. E. Batista, "Speeding up all-pairwise dynamic time warping matrix calculation," in Proceedings of the 2016 SIAM International Conference on Data Mining. SIAM, 2016, pp. 837–845.
[67] S. Salvador and P. Chan, "FastDTW: Toward accurate dynamic time warping in linear time and space," Intelligent Data Analysis, vol. 11, no. 5, pp. 561–580, 2007.
[68] E. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2000, pp. 285–289.
[69] S. Chu, E. Keogh, D. M. Hart, M. J. Pazzani , "Iterative deepening dynamic time warping for time series." in SDM. SIAM, 2002, pp. 195–212.
[70] A. Sharabiani, H.Darabi, A.Rezaei, S.Harford, H.Johnson and F. Karim, "Efficient Classification of Long Time Series by 3-Dimensional Dynamic Time Warping." IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47 Issue: 8. doi:10.1109/TSMC.2017.2699333.
[71] Makonin, S. et al. Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014. Sci. Data 3:160037 doi: 10.1038/sdata.2016.37 (2016).
[72] D. Murray and L. Stankovic. Refit: Electrical load measurements. http://www.refitsmarthomes.org/.
[73] A. Sharabiani, H.Darabi, A.Rezaei, S.Harford, H.Johnson and F. Karim,"Efficient Classification of Long Time Series by 3-Dimensional Dynamic Time Warping." IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47 Issue: 8. doi:10.1109/TSMC.2017.2699333.
[74] "A Novel Bayesian and Chain Rule Model on Symbolic Representation for Time Series Classification ", In Automation Science and Engineering (CASE), IEEE International Conference on, pp. 532-537. IEEE, 2016.
[75] A.Sharabiani, H.Darabi, S.Harford, H.Johnson, F. Karim, E.Douzali and S.Chen, "Asymptotic Dynamic Time Warping Calculation with Utilizing Values Repetition", Knowledge and Information System Journal, 2017, In press.

APPENDICES

APPENDIX A

8/24/2017 RightsLink® by Copyright Clearance Center

Copyright Clearance Center RightsLink®

Home | Create Account | Help

IEEE Requesting permission to reuse content from an IEEE publication

Title: Efficient Classification of Long Time Series by 3-D Dynamic Time Warping
Author: Anooshiravan Sharabiani
Publication: Systems, Man, and Cybernetics: Systems, IEEE Transactions on
Publisher: IEEE
Date: Dec 31, 1969
Copyright © 1969, IEEE

LOGIN
If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

APPENDIX B

Copyright Transfer Statement

The copyright to this article is transferred to Springer-Verlag (respective to owner if other than Springer-Verlag and for U.S. government employees: to the extent transferable) effective if and when the article is accepted for publication. The copyright transfer covers the exclusive right to reproduce and distribute the article, including reprints, translations, photographic reproductions, microform, electronic form (offline, online) or any other reproductions of similar nature.

An author may make his/her article published by Springer-Verlag available on his/her home page provided the source of the published article is cited and Springer-Verlag is mentioned as copyright owner. Authors are requested to create a link to the published article in Springer's internet service.
The link must be accompanied by the following text:
"The original publication is available at http://www.springerlink.com."

Please use the appropriate URL and/or DOI for the article. Articles disseminated via SpringerLink are indexed, abstracted and referenced by many abstracting and information services, bibliographic networks, subscription agencies, library networks, and consortia.

The author warrants that this contribution is original and that he/she has full power to make this grant. The author signs for and accepts responsibility for releasing this material on behalf of any and all co-authors.
After submission of this agreement signed by the corresponding author, changes of authorship or in the order of the authors listed will not be accepted by Springer-Verlag.

Journal: Knowledge and Information Systems Editorial Manuscript Number: KAIS-D-16-00786R3

Title of article: Asymptotic Dynamic Time Warping Calculation with Utilizing Values Repetition

Author(s): Anooshiravan Sharabiani; Houshang Darabi; Samuel Harford; Elnaz Douzali; Fazle Karim; Hereford Johnson; Shun Chen

Author´s signature: A.Sharabiani

VITA

Anooshiravan Sharabiani

SEL 4209, 842 W Taylor, Chicago, IL 60607. Phone: (312) 547 0360
Email: ashara3@uic.edu

Education

University of Illinois at Chicago (UIC) Chicago, IL (Jan. 2013- Sept.2017)

Ph.D. Industrial Engineering and Operations Research

Sharif University of Technology Tehran, Iran (Sept. 1999- May. 2001)

MSc. Industrial Engineering

University of Science and Technology Tehran, Iran (Sept. 1995- May. 1999)

BSc. Industrial Engineering

Qualifications and Skills

- More than 10 years work experience in large organizations as a data analyst and more than 4 years recent work experience as a data scientist.
- Extensive knowledge in Machin Learning, Data Mining and Time Series Analysis.
- A strong quantitative background in math and analytical skills including statistics, simulation, predictive modeling, sensor data analysis and optimization.

- Expert in advance MS. Excel and data visualization tools / techniques (Plotly, R ggplot2, Tableau)
- Extensive knowledge and experience in databases, SQL, R and Python programming. Experienced in problem definition, data assembling, data cleaning and transformation, data visualization, conducting research, experiments, and performing advanced statistical and data mining modeling.

Professional Experience

Data Scientist, Research/ Teaching Assistant Chicago, IL (Jan. 2013- Present)

University of Illinois at Chicago

- Developed novel algorithms to improve scalability and accuracy of Time Series Classification (TSC) techniques. The performance of the proposed algorithms were compared against the state of the art models in the literature on more than 85 benchmark datasets and their outperformance, both in terms of accuracy and speed were proven.
- Created a predictive model to detect seizure occurrences at least 10 minutes in advance. Implemented efficient methods to import the data of epileptic patients' EEG (500 GB) from 6 installed sensors on the patients' brain and applied different machine learning methods using R to develop the prediction model which resulted in 75% prediction accuracy.
- Created a dynamic forecasting system for the College of Engineering at UIC to predict the required class capacity of each engineering course for the next two semesters. A macro-level approach (analyzing the historic trends of class registration) and a micro-Level

approach (exploring every individual student's enrollment patterns and predicting his/her future enrollment decisions) were examined for developing the forecasting system and finally a hybrid model was developed which resulted in more than 85% prediction accuracy.

- Created a visualization tool to demonstrate the behavior and pattern of students in their study paths. Detected 3 clusters of students based on their behavior on their study path. Then analyzed the characteristics of these clusters like GPA of students, type of students, etc. in each cluster. Created prediction models to predict ahead of time what the study path of a specific student would be in coming semesters in two steps.
- Predicted the visit pattern and delivery location of Medicaid obstetric patients at the University of Illinois Hospital & Health Sciences System by developing a novel trace-based system using the patients' sensitive data. The data set contained the patients' demographics, medical history, visit patterns, and medical codes in each visit for the last 10 years. The developed model resulted in 76% prediction accuracy.
- Created a web-based information management system to prepare a cost estimate for research projects, based on the required services from a clinical service provider.

Lecturer, Chicago, IL (Aug. 2015- Dec. 2015)

University of Illinois at Chicago

- Instructed an undergraduate-level course on Data Mining and Programming in R to 90 undergraduate students from the College of Engineering. The course materials contained introductory and advanced methods and techniques in statistical analysis and Machine Learning using R.

Senior Data Analyst, Tehran, Iran (Nov. 2008- Dec. 2012)

Fulmen Company

- Designed and implemented a reporting system to inform top management level the key status of all projects including: marketing-tendering activities, scope change, risk management, and the financial condition of the company, including the cash flow of each project within the organization.
- Devised an incentive system to distribute a performance-based profit/gain sharing method throughout the entire company.
- Formulated a model to optimize the total number of employees the company should hire in each department due to a dynamic, exponential growth in client contracts.
- Leveraged the structure of CRM best practices to cultivate and deploy software which enhanced and regulated the procurement process.
- Designed and installed a user-friendly control tool in MS Access which tracked action items and due dates from senior executive board meetings to control the accountability of senior management; no action items were missed after the implementation of this program.

Data Analyst | Project planning and Control Manager, Tehran, Iran (July. 2003- Oct. 2008)

Fulmen Company

- Developed a project control system which synthesizes daily input data to provide periodic Projects Review Reports which include the progress and financial situation of the projects.
- Managed the ideation, development, and implementation of a software to detect delays, their causes along with their responsible sources, and prepared analytic reports on the effects of each delay over the project.

- Developed an analytical tool to evaluate live tender information which resulted in a dynamic competitive strategies which increased the likelihood of winning new contracts; post implementation win-rate increased by 88%.

Data Analyst, Tehran, Iran (May. 2001- June. 2003)

Tehran Padena Company

- Designed and developed a Material Requirements Planning system linked to the factory's inventory control software including production planning and control, ordering system, alarming system of shortage in inventory (materials and products), lot sizing and monitoring the production capacities.
- Developed the Business Plan of organization including demand forecasting of every products, Master production schedule (MPS), resource requirements, purchasing materials & components planning, sale forecasting, overall costs estimating and financial analysis of organization.
- Served as direct head of the leadership team; established a quality assurance system throughout the organization.
- Designed and implemented a subcontractor's evaluation system and developed a subcontractor qualification process base on the delivery due dates and quality & quantity criteria.
- Reviewed and revised the incoming material inspection system and prepared progress reports on suppliers' contracts.

Publications

Journal Papers

- "High-Speed 3-Dimensional Dynamic Time Warping for Classification of Long Time Series", IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017.
- "Asymptotic Dynamic Time Warping Calculation with Utilizing Values Repetition" Accepted by Knowledge and Information System, 2017.
- "A Framework for Accurate Time Series Classification" under review by Knowledge and Information System, 2017.

Conference Papers

- "A Novel Bayesian and Chain Rule Model on Symbolic Representation for Time Series Classification ", In Automation Science and Engineering (CASE), IEEE International Conference on, pp. 532-537. IEEE, 2016.
- "An Enhanced Bayesian Network Model for Prediction of Students' Academic Performance in Engineering Programs", IEEE Global Engineering Education Conference in Turkey, 2014.
- "Gain-Sharing Systems in Project-Based Organizations" proceeding of 6th International Project Management Conference in Iran, 2010.
- "Income, Cash and Cost Control Using Earned Value Systems" proceeding of 4th International Project Management Conference in Iran, 2008.
- "On-time Project Delay Analysis" proceeding of 3rd International Project Management Conference in Iran, 2007.