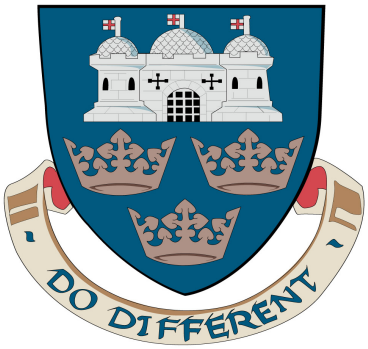
Time Series Classification through Transformation and Ensembles

Jason Andrew Lines

A Thesis Submitted for the
Degree of Doctor of Philosophy



University of East Anglia
School of Computing Sciences

February 2015

Abstract

The problem of time series classification (TSC), where we consider any real-valued ordered data a time series, offers a specific challenge. Unlike traditional classification problems, the ordering of attributes is often crucial for identifying discriminatory features between classes. TSC problems arise across a diverse range of domains, and this variety has meant that no single approach outperforms all others.

The general consensus is that the benchmark for TSC is nearest neighbour (NN) classifiers using Euclidean distance or Dynamic Time Warping (DTW). Though conceptually simple, many have reported that NN classifiers are very difficult to beat and new work is often compared to NN classifiers. The majority of approaches have focused on classification in the time domain, typically proposing alternative elastic similarity measures for NN classification. Other work has investigated more specialised approaches, such as building support vector machines on variable intervals and creating tree-based ensembles with summary measures.

We wish to answer a specific research question: given a new TSC problem without any prior, specialised knowledge, what is the best way to approach the problem? Our thesis is that the best methodology is to first transform data into alternative representations where discriminatory features are more easily detected, and then build ensemble classifiers on each representation. In support of our thesis, we propose an elastic ensemble classifier that we believe is the first ever to significantly outperform DTW on the widely-used UCR datasets. Next, we propose the shapelet-transform, a new data transformation that allows complex classifiers to be coupled with shapelets, which outperforms the original algorithm and is competitive with DTW. Finally, we combine these two works with with heterogeneous ensembles built on autocorrelation and spectral-transformed data to propose a collective of transformation-based ensembles (COTE). The results of COTE are, we believe, the best ever published on the UCR datasets.

Acknowledgements

First and foremost I would like to thank my supervisor, Dr. Anthony Bagnall, and my family. Without Tony's invaluable advice and guidance, and the continued support and encouragement from my parents and close family, the work in this thesis would not have been possible.

I would like to thank my examiners, Prof. Niall Adams and Dr. Beatriz de la Iglesia, for their patience and insight while examining this thesis. I would also like to thank Tony's other PhD students during my study, Jon and Luke, and my second supervisor, Dr. Richard Harvey. Additional thanks go to my peers at UEA, particularly those who I shared a lab with for four years (and everyone in the graphics lab since I spent as much time in there too!). I'd also like to thank all of the staff and students in the School of Computing Sciences at UEA who have supported me throughout my studies. Special mentions go to Felix and Dom for being a constant source of entertainment and distraction, Ollie for providing the cake and enough surreal moments to last a lifetime, Luke for his 80's pop-rock outbursts, and everyone from CMP and ITCS that played football on Thursdays.

Finally, I'd like to thank some important people outside of university. Thank you to Shane Migliore and everyone at Apple for helping make Austin my home for four months at the end of my PhD. As well as new friends, I'd like to thank those who have always been there for me. In particular, thank you to Bedford, Beth, Danny, David, Emma, Faires, James, Jim, and Sarah. Without you, I would have gone crazy a long time ago. Also, thank you to Freddie and Ollie (my dogs) for not eating my thesis, and a special thank you to my mother for painstakingly helping me with the final round of proofreading!

In loving memory of Kenneth Thomas Deare.

Contents

List of Publications

As First Author

- Jason Lines, Anthony Bagnall, Patrick Caiger-Smith, and Simon Anderson. Classification of household devices by electricity usage profiles. In *Intelligent Data Engineering and Automated Learning-IDEAL 2011*, pages 403–412. Springer Berlin Heidelberg, 2011.
- Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 289–297. ACM, 2012.
- Jason Lines and Anthony Bagnall. Alternative quality measures for time series shapelets. In *Intelligent Data Engineering and Automated Learning-IDEAL 2012*, pages 475–483. Springer Berlin Heidelberg, 2012.
- Jason Lines and Anthony Bagnall. Ensembles of elastic distance measures for time series classification. In *Proceedings of the 14th SIAM International Conference on Data Mining (SDM)*, pages 524–532. 2014.
- Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, pages 1–28, 2014.

As Co-author

- Anthony Bagnall, Luke M Davis, Jon Hills, and Jason Lines. Transformation based ensembles for time series classification. In *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*, pages 307–318. SIAM, 2012.
- Luke M Davis, Barry-John Theobald, Jason Lines, Andoni Toms, and Anthony Bagnall. On the segmentation and classification of hand radiographs. *International Journal of Neural Systems*, 22(05), 2012.
- Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.

Chapter 1

Introduction

Time series data, which we consider as any real-valued ordered data, arise across many domains. These include, but are not limited to: econometrics, medicine, weather, motion capture, image processing, computational biology, signal processing, and pattern recognition. The problem of time series classification (TSC) is a specialisation of the more general classification problem; the objective of classification is, given a new test instance of data, can the category, or *class*, of this observation be determined from characteristics that have been extracted from a set of previously observed training data with known class labels?

To an extent, all classification problems rely on identifying explanatory features within the data, then using a measure of similarity to quantify the relationships between them to inform the decision process. TSC offers a specific challenge, as the ordering of the data may be crucial in discriminating between class values. For example, the data may have an embedded structure, such as autocorrelation, or trends. To this end, traditional classification approaches may not be best suited to TSC problems. This has prompted many different approaches for solving TSC problems to be proposed in the literature [9, 58, 20, 49, 107, 80, 106, 50, 55]. These range from the most generic end of the spectrum (using standard classification algorithms on time series data and ignoring the dependency between attributes), to the most specialised (creating bespoke problem-specific solutions). There are also many other solutions in between, such as using support vector machines built on variable intervals [93], or tree-based ensembles built on summary measures [34].

However, it has been observed many times in the literature that, in general, a nearest neighbour classifier using a measure of time series similarity is very difficult to beat. This is noted succinctly by Batista *et al.* [10], who state that '*there is a plethora of classifica-*

tion algorithms that can be applied to time series; however, all of the current empirical evidence suggests that simple nearest neighbour classification is very difficult to beat'. One of the simplest approaches for solving TSC problems is to to use a one-nearest neighbour (1-NN) classifier with Euclidean distance. However, the widely-accepted benchmark in TSC currently is to use a 1-NN classifier coupled with Dynamic Time Warping (DTW) with a warping window set through cross-validation (CV) [21, 47, 34].

1.1 Motivation

Given a new TSC problem to solve, the optimal solution will most likely be achieved by creating a custom, bespoke solution that is tailor-made to the problem at hand. However, this approach is typically very time and resource consuming, and will likely result in a solution that is very difficult (or impossible) to generalise to other problems. The *no free lunch* theorem [105] applies to many fields, and TSC is not immune; there is no single solution that is optimal for all problems. As already discussed, the literature argues that, in general, nearest neighbour classification is *very difficult* to beat, and DTW with a 1-NN classifier is considered the current gold-standard for TSC. With this in mind, when presented with a new TSC problem the sensible approach would be to build a NN classifier with DTW. However, we believe this approach is naïve.

Firstly, there are many different types of similarity that can be observed between time series data. For example, series could change at similar points in time (time-based similarity), or they could have similar underlying curves or trends (global shape-based similarity). Also, the series could be similar in the ways that they change due to their internal structures, such as the autocorrelation of the series and how the values of subsequent readings are influenced by previous readings (change-based similarity). Additionally, the presence of localised common subsequences could be indicative of class membership. An approach using 1-NN and DTW would be well suited to measuring time-based similarity, but may struggle to identify the best discriminatory features in the other three cases. It would therefore be desirable if we could determine *a priori* which type of similarity should be used for a new problem.

Secondly, even if using a simple 1-NN approach, there are many alternative similarity measures that have been proposed in the literature that could be used in place of DTW. Many are claimed to be at least as effective as DTW, and in some cases it is stated that the alternatives are more effective. However, evaluation methods are inconsistent and are sometimes based on simple head-to-head results and anecdotal evidence. It is currently unclear whether any alternatives are truly more effective than DTW, but it is clear that

various alternatives capture similarity differently to DTW (for example, edit distance-based approaches [25, 26, 79]). It would be desirable to know in advance whether DTW is the most appropriate measure to use with a 1-NN classifier for a given problem, or even if it is possible to combine the output of multiple measures to make a more diverse and informed decision. This leads to the research question that has influenced the work throughout this thesis: *given a new TSC problem with no prior, specialised knowledge, what is the best way to approach classification?*

In [3], it was shown that a simple way to gain improvement in TSC problems is to transform data into alternative domains where where discriminatory features are more easily detected. We use this as a starting point; our thesis is that the best way to approach a TSC problem without any prior specialised knowledge is to first transform data into alternative representations where discriminatory features are more easily detected. This would potentially allow algorithms to not only measure similarity between series in the time domain, but also consider similarity in change, global-shape, and local-shape. Then, we believe that through using transparent ensemble schemes, we can build ensemble classifiers in each of these domains to form constituent ensembles in a diverse collective of transformation-based ensembles. Many algorithms embed data transformation within classification algorithms. We believe that by transforming data independently of classifiers, and by utilising simple ensemble voting schemes, we can produce a very accurate and transparent classifier that can outperform the benchmark set by DTW 1-NN, and provide a framework that can easily be extended in future work to add additional classification algorithms and data representations.

1.2 Contributions

In pursuit of providing support for our thesis, numerous experiments were carried out and novel algorithms were proposed. The main contributions of this thesis are as follows:

- **Time series classification in the time domain**. An extensive study was carried out using 75 datasets to evaluate whether DTW with warping set through cross-validation (DTWCV) is still the benchmark for TSC. Initially this investigation focused on whether DTW was *hard to beat* through a comparative study between various standard classification algorithms and 1-NN classifiers implemented using Euclidean distance and DTW with a full window. After establishing that none of these classifiers significantly outperformed DTW, an investigation was carried out to determine the best configuration for DTW with NN classifiers, including whether

the neighbourhood size of the NN classifier or window width of the DTW measure should be set through cross-validation. After answering these questions and recommending that DTW be implemented with a warping window set through cross-validation and one nearest neighbour, we carried out a comparison of DTWCV to alternative elastic similarity measures that have been proposed in the literature, including approaches such as Weighted DTW [58], Move-Split-Merge [101], and Time Warp Edit Distance [79]. Over the 75 datasets tested, we found that no alternative significantly outperformed DTWCV, concluding that DTWCV with 1-NN was still the benchmark in TSC. However, through investigating the alternative elastic measures, we demonstrated that while the measures did not produce significantly different accuracies, they did produce significantly different predictions. This motivated an investigation into using simple ensemble schemes to build an ensemble classifier with each of the elastic measures. The resulting classifier, the elastic ensemble (EE), is significantly more accurate than DTWCV over the 75 datasets. Additionally, the EE is significantly more accurate than DTW on the 46 UCR datasets that are widely throughout the TSC literature for evaluating new algorithms. This result is noteworthy, as we believe this is the first time a classifier has ever outperformed DTWCV on the UCR TSC problems. This work is reported in Chapter 5 and was published in [75, 76].

- **A novel shapelet transform**. As outlined in our thesis, we believe that the best approach for TSC is to evaluate similarity in additional domains to the time domain. One area of time series similarity that is currently under represented in the literature is similarity in local-shape. A recent approach, time series shapelets [107], was proposed to match series according to common local patterns through extracting discriminatory subsequences to build a decision tree classifier. This approach was designed to produce intuitive results, but restricts shapelets by embedding them within a classifier. We use this as a starting point to create a new time series transformation that mitigates the limitations of a shapelet decision tree approach by extracting the top k shapelets from a dataset in a single pass through using a novel caching algorithm. The extracted shapelets are used to transform series into a new k-dimensional representation, where each feature is the distance between a shapelet and the input series. The result of this transform is that the new representation can be applied to any standard classification algorithm, facilitating classification based on local shape-based similarity. We demonstrate that using the shapelet transform with popular classification algorithms such as support

vector machines and random forest significantly outperforms the original shapelet tree-based approach, and we also provide information to show that classification accuracy can be greatly improved for problems that are not well-suited to the time domain when compared against DTWCV. We also extend our work to consider alternative measures of shapelet quality that are more suited to the transformation approach, and demonstrate that they are significantly faster for shapelet extraction than the information gain measure used in the original approach. Finally, we produce a case study using a popular UCR dataset to mirror the case study provided in the original shapelet work to demonstrate that our shapelet transform retains the intuition provided by the original approach, and may also allow for greater insight to be gained. This work is described in Chapter 6 and was published in [78, 74, 52].

- **COTE: The collective of transformation-based ensembles for time series classification**. This work builds upon the conclusion in [3] that the simplest way to improve accuracy for TSC problems is to transform data into alternative domains where discriminatory features are more easily detected. This is extended by creating a collective of ensemble classifiers built in four domains: time, local shape, global shape, and change. We use the elastic ensemble proposed in Chapter 5 for time domain classification, and we create a simple heterogeneous ensemble to apply to transformed data in the remaining three domains. For representing local shape-based similarity, we transform series using the shapelet transform that we propose in Chapter 6. For global shape-based similarity we transform data using the power spectrum, and for change, we use autocorrelation-based transforms. Through extensive experimentation on 72 datasets, including all of the 46 UCR datasets, we demonstrate that the simple collective formed by including all classifiers in one ensemble is significantly more accurate than any other previously published TSC algorithm, including the EE. We call this classifier the collective of transformation-based ensembles (COTE), and the results provided by COTE provide strong support for our thesis. Finally, after proposing and testing the initial configuration for COTE, we propose alternative versions using weighting and selection schemes in pursuit of adding extra insight to results without reducing accuracy. This work is described in Chapter 7, and the results of the COTE are, we believe, the best ever published on the UCR datasets.

1.3 Thesis Organisation

The remainder of this thesis is organised as follows. In Chapter 2, a thorough review of the TSC literature is carried out, including specific emphasis on nearest neighbour classification in the time domain with DTW and other alternative elastic distance measures. In Chapter 3 we introduce the datasets that are used throughout this thesis, including new problems that we have provided and shared with the TSC community for the first time. Chapter 4 discusses transformation approaches for time series to allow for similarity to be assessed in alternative domains, while Chapter 5 focuses specifically on the time domain and culminates in the proposal of the EE. Chapter 6 moves away from time domain similarity to propose a novel transform to capture similarity in local-shape: the shapelet transform. The findings in Chapter 5 and 6 are then combined in Chapter 7 where the COTE is proposed and tested, demonstrating that forming a collective of ensemble classifiers built on different time series representations can significantly outperform DTWCV, and any other classifier that we know of, across our test set of 72 datasets and the UCR data. Finally, in Chapter 8, we conclude this thesis by summarising the contributions of this work and discussing possible future direction.

Chapter 2

Technical Background and Related Work

This chapter introduces the relevant background materials for this thesis. We motivate and introduce the problem of time series classification (TSC), and present a review of the leading solutions that have been proposed in the literature. Specifically, we focus on the common benchmark technique of using simple nearest neighbour (NN) classifiers with Euclidean distance Dynamic Time Warping (DTW) to solve the problem in the time domain, and explore potential alternative elastic measures that can be combined with NN classifiers. Following this, we investigate more complex solutions that have been introduced in the literature, including approaches built on transforming data into other domains and combining decisions of multiple classifiers.

2.1 Time Series Classification

Many problems exist in the time series data mining literature, including classification, clustering, indexing, and querying. In this thesis, we focus solely on the problem of time series classification (TSC). We define a time series as a sequence of data that is typically recorded in temporal order at fixed time intervals. For the problem of TSC, we use a set of n time series

$$T = \{T_1, T_2, ..., T_n\} \tag{2.1}$$

where each series consists of m real-valued ordered observations

$$T_i =< t_{i,1}, t_{i,2}, ..., t_{i,m} > \tag{2.2}$$

and a class value c_i. Given T, the TSC problem is to find a function that maps from the space of possible time series to the space of possible class values. For simplicity, in our definition we assume that all series of T are the same length.

A key feature that distinguishes TSC problems from general classification problems is that the features within the data that discriminate between class values are often embedded within the inter-relationship structure of the data. In generic classification problems, attributes of the data are often independent; in TSC problems, the ordering and structure of the data often play a crucial role in uncovering features that define class relationships.

The nature of TSC problems means that the representation of the data is a crucial part of any TSC algorithm, and all TSC approaches rely to an extent on measures of similarity between data. There are three broad categories of TSC similarity that appear in the literature:

- **Similarity in time** can be observed when series from a class are observations of an underlying common curve in the time dimension. When there is no noise in the observation, correlation-based measures or the Euclidean distance can be used to effectively measure similarity between series. When there is noise in the time dimension or slight phase-shift, elastic distance measures can be used. Dynamic time warping (DTW) is by far the most popular of such measures in the literature [62, 91, 35, 61, 64, 90, 49].

- **Similarity in shape** is when class membership is characterised by a common shape in the data that is phase-independent. This can cover two such scenarios: firstly, if the common shape involves the whole series, but the shape is shifted between instances of the same class, transforming the data into the frequency domain can uncover discriminating features [22, 57]. Secondly, if the common shape is local and embedded within instances of the same class, subsequence techniques such as shapelets [107, 108, 80, 89] can measure shape-based similarity accurately without being affected by noise throughout the rest of the series.

- **Similarity in change** is where the discriminatory features of a dataset are embedded within the autocorrelation structure of the data. This type of similarity is the least covered in the literature, but can be employed for TSC by transforming series into the change domain through the use of autocorrelation-based transforms, or by applying autoregressive moving average models (ARMA) to the data and judging similarity on model parameters [27, 4, 5].

In this chapter we survey and review the TSC literature across each of these types of similarity. Similarity in time is by far the most represented of the three in the TSC literature, and this is reflected by the extensive discussion of nearest neighbour classification with alternative elastic similarity measures in Section 2.3. A discussion of popular TSC algorithms follows in Section 2.4, including approaches that involve transforming data into other domains to uncover similarity. In Section 2.5, the process of combining multiple algorithms to form ensemble classifiers is discussed and motivated with an example of a heterogeneous ensemble in the time domain. Before exploring specific TSC algorithms however, we must introduce a methodology for consistently comparing classifiers across many datasets.

2.2 Comparing Classifiers

In order to test the thesis outlined in Chapter 1, it is necessary to objectively compare classification algorithms over many datasets. In the literature it is common to base support for new algorithms on anecdotal evidence or simple win/loss statistics. We wish to be more thorough in our analysis during this thesis, and adopt the procedure outlined in [33] to test for statistical significance between classifiers. The approach is based on a two-stage rank-sum test using the non-parametric equivalent to analysis of variance (ANOVA).

The first stage of the approach tests the null hypothesis that there is no significant difference between the average ranks of k classifiers on n datasets, against the alternative hypothesis that at least one classifier's mean rank is different. Given M, the k by n matrix of classification accuracies where $m_{i,j}$ is the accuracy of the i^{th} classifier on the j^{th} dataset,

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{k,1} & m_{k,2} & \cdots & m_{k,n} \end{bmatrix} \tag{2.3}$$

the first step is to calculate the corresponding n by k matrix R, where $r_{i,j}$ is the rank of the i^{th} classifier on the j^{th} problem, and the ranks of classifiers with equal error are

averaged:

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{k,1} & r_{k,2} & \cdots & r_{k,n} \end{bmatrix} \tag{2.4}$$

From R the average rank of classifier j is calculated as $\bar{r_j} = \frac{\sum_{i=1}^{n} r_{i,j}}{n}$. To test the hypothesis, the Friedman statistic Q,

$$Q = \frac{12n}{k(k+1)} \cdot \left[\sum_{j=1}^{k} \bar{r_j}^2 - \frac{k(k+1)^2}{4} \right] \tag{2.5}$$

can be approximated using a Chi-squared distribution with $(k-1)$ degrees of freedom to test the null hypothesis that there is no difference between the mean ranks of the classifiers. However in [33], Demšar notes that this calculation is often conservative, and proposes using the following statistic:

$$F = \frac{(n-1)Q}{n(k-1) - Q}, \tag{2.6}$$

that, under the null hypothesis, follows an F distribution with $(k-1)$ and $(k-1)(n-1)$ degrees of freedom. If the result of this calculation is that we can reject the null hypothesis, resulting in at least one classifier having a significantly different average rank, the second stage of the approach proposed by Demšar involves performing *post-hoc* pair-wise Nemenyi tests to observe where differences occur. The test states that the average ranks of two classifiers are significantly different if they differ by at least the *critical difference*, calculated as:

$$CD = q_a \sqrt{\frac{k(k+1)}{6n}}, \tag{2.7}$$

where q_a is based on the studentised range, where the difference between the largest and smallest values in the sample is measured in units of standard deviation. By comparing all classifiers in this way, Demšar proposes that a critical difference diagram can be created to effectively summarise the results. This is formed by creating a diagram where the average ranks for each classifier are labelled on a numerical range, and classifiers that are not significantly different from one another are organised into *cliques*. A clique is represented by a solid black line, and allows for simple interpretation of results by

observing whether classifiers are within a common clique. If two classifiers do not belong to at least one common clique, the average ranks of the classifiers are significantly different. A motivational example of a critical difference diagram is presented in Figure 2.1. We use critical difference diagrams throughout this thesis to compare results of multiple classifiers over multiple datasets.

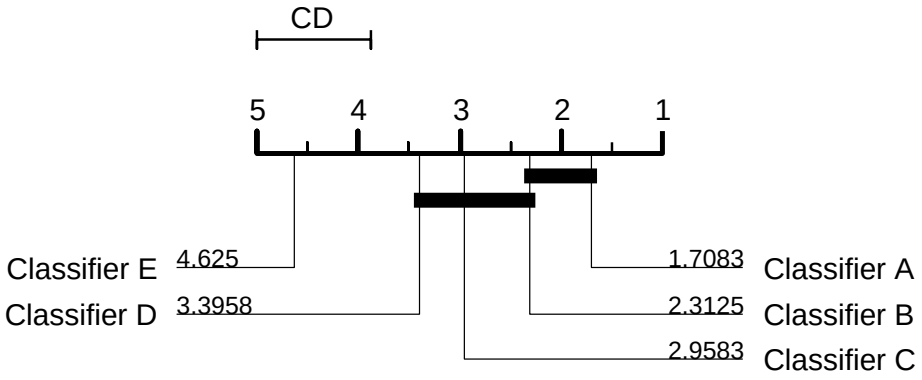


Figure 2.1: An example of a critical difference diagram with five fictional classifiers. The critical difference in this diagram is 1.1228; The average ranks of Classifier A and Classifier B do not differ by more than the critical difference, so the ranks are not significantly different. However, Classifier A and Classifier C are not within the same clique, so it is demonstrated that they are significantly different.

2.3 Nearest Neighbour Classification in the Time Domain

Numerous algorithms and approaches have been proposed in the literature for solving TSC problems. However, as eluded to in Chapter 1, currently there has been no conclusive evidence reported to suggest that any technique is significantly more accurate than using a simple nearest neighbour (NN) classifier for TSC. This is reinforced by Batista *et al.* in [9], who state that '*there is a plethora of classification algorithms that can be applied to time series; however, all of the current empirical evidence suggests that simple nearest neighbor classification is very difficult to beat*'. The sentiment of this statement is echoed in various other work throughout the literature, with the general consensus that NN classifiers combined with either Euclidean distance or Dynamic Time Warping are the current benchmark for TSC [21, 47, 34].

2.3.1 *k*-Nearest Neighbour (*k*-NN) Classifiers

The k-NN classifier is one of the oldest, simplest, and most commonly used classification algorithms that is fundamental to many TSC approaches, particularly approaches in the time domain (for example, [60, 92, 106, 79, 9, 21, 47, 101]). The classifier was first documented in [39] and is intuitively very simple to implement for TSC problems. Given a set of labelled training time series T and a query series q with an unknown class value, the objective is to find the k most similar series to q in T, where $1 \leq k \leq n$. The predicted class value c^* for q is selected as the modal value of the k nearest neighbours. The value for k is often set as an odd number to reduce the chance of ties, but in cases where there is no majority, ties are split randomly.

A fundamental part of the k-NN algorithm is how similarity is measured between q and the instances in T. This topic has arguably generated the most interest in the literature, typically combining measures with one nearest neighbour classifiers (1-NN). Early work used Euclidean distance and other L_p-norms for assessing similarity until dynamic time warping (DTW) was first applied to time series data [12]. DTW has been held as the benchmark in TSC ever since. Given a single TSC problem, the best absolute accuracy is likely to ultimately be provided by a bespoke, one-of-a-kind solution that is tailored to a specific problem. For TSC in general however, to the best of our knowledge, there is no documented evidence of any algorithm outperforming DTW with 1-NN. Therefore much of the work that initially followed the introduction of DTW focused primarily on how to speed-up the similarity measure, and later research investigated how placing limitations on the amount of warping allowed in the DTW search could lead to improved accuracy. Recent research has focused on extensions to the original DTW approach (for example, using derivatives [63] and weightings [58]), while numerous alternative elastic measures have also been proposed (such as edit distance-based and hybrid measures [79, 101]).

2.3.2 Euclidean Distance

The Euclidean distance is one of the simplest similarity measures available for comparing time series. Given two series $\mathbf{a} =< a_1, a_2, ..., a_m >$ and $\mathbf{b} =< b_1, b_2, ..., b_m >$, the Euclidean distance $d_{Euclid}(a, b)$ is given as the square root of the sum of squares of the differences between attributes in each series:

$$d_{Euclid}(a, b) = \sqrt{\sum_{i=1}^{m}(a_i - b_i)^2}. \tag{2.8}$$

A common speed-up of the calculation is to to omit the square root. However, since the Euclidean distance requires no parametrisation, there is little that can be done to the measure specifically to improve performance or speed since originally introduced in early work such as [29]. Researchers have used related approaches, such as the Minkowski distance [2] or Mahalanobis distance [22], but Euclidean distance has remained the most popular of these measures. Therefore research has often focused on speeding up applications that involve the use of the distance measure instead. A common theme in the literature has been to devise a novel approach towards representing time series data, then using Euclidean distance on representations of the data with a lower dimensionality. For example, [1] uses a Fourier transformation to create representations with fewer attributes, accelerating distance calculations between series using the Euclidean distance. In [23], a similar approach is applied using wavelets, and in [59] the authors use a multi-resolution approach with wavelets to calculate similarity between series using Euclidean distance. Such approaches have lead to a spate of approximation methods appearing in the literature, with basis in topics such as spectral approaches, wavelets, piece-wise approximations, and symbolic representations (see Section 2.6.2 for more details).

2.3.3 Dynamic Time Warping

Dynamic Time Warping (DTW) is commonly used as a measure of similarity between series in the time domain. DTW is popular in the literature as unlike the Euclidean distance, the measure has elastic properties that allow it to mitigate against distortions in the time axis [90].

Given two time series $\mathbf{a}$ and $\mathbf{b}$ that we wish to compare, let $M(\mathbf{a}, \mathbf{b})$ be the $m \times m$ point-wise distance matrix between $\mathbf{a}$ and $\mathbf{b}$, where $M_{i,j} = (a_i - b_j)^2$. A warping path

$$P =< (e_1, f_1), (e_2, f_2), \ldots, (e_s, f_s) > \tag{2.9}$$

is a set of points (i.e. pairs of indexes) that define a traversal of M. So, for example, the Euclidean distance $d_E(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{m}(a_i - b_i)^2$ is the path along the diagonal of M.

A valid warping path must satisfy the conditions $(e_1, f_1) = (1, 1)$ and $(e_s, f_s) = (m, m)$ and that $0 \leq e_{i+1} - e_i \leq 1$ and $0 \leq f_{i+1} - f_i \leq 1$ for all $i < m$.

The DTW distance between series is the path through M that minimizes the total distance, subject to constraints on the amount of warping allowed. Let $p_i = M_{a_{e_i}, b_{f_i}}$ be the distance between elements at position e_i of $\mathbf{a}$ and at position f_i of $\mathbf{b}$ for the i^{th} pair of points in a proposed warping path P. The distance for any path P is

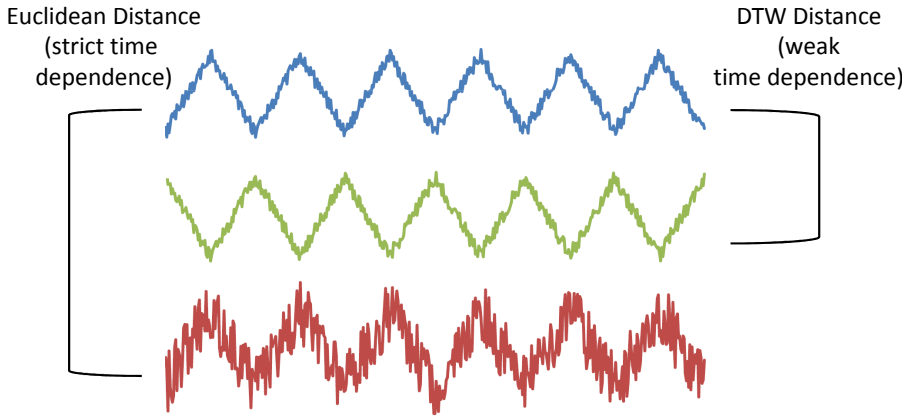


Figure 2.2: An example to demonstrate the elastic properties of DTW. The first two time series have identical values but are out of phase, so the Euclidean distance between them is large as the peaks and troughs do not align. The elasticity of DTW mitigates this however and finds that the first two series are the most similar.

$$D_P(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{s} p_i. \tag{2.10}$$

If $\mathcal{P}$ is the space of all possible paths, the DTW path P^* is the path that has the minimum distance, i.e.

$$P^* = \min_{P \in \mathcal{P}}(D_P(\mathbf{a}, \mathbf{b})), \tag{2.11}$$

and hence the DTW distance between series is

$$D_{P*}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{k} p_i. \tag{2.12}$$

The optimal path P^* can be found exactly through dynamic programming, but this can be a time consuming operation. Therefore numerous speed-ups have been proposed to place a restriction on the amount of possible warping that is allowed. The most commonly used approach is to use the Sakoe-Chiba band [95] that was originally proposed in the speech processing literature. This restriction is equivalent to putting a maximum allowable distance between any pairs of indexes in a proposed path. If the warping window, r, is the proportion of warping allowed, then the optimal path is constrained so that

$$|e_i - f_i| \leq r \cdot m \quad \forall (e_i, f_i) \in P^*. \tag{2.13}$$

A popular alternative to the Sakoe-Chiba band is Itakura's Parallelogram [56], which also originated in speech research. The parallelogram approach imposes a similar restriction on warping, but instead favours smaller warpings at the beginning and end of $P*$ while allowing more freedom mid-path.

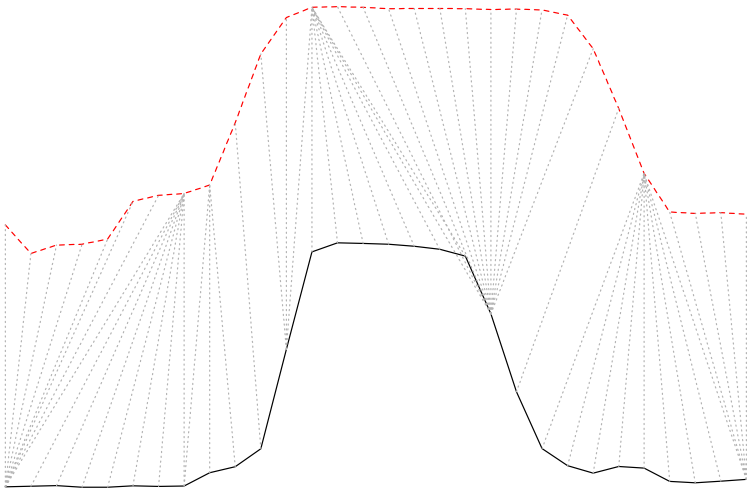


Figure 2.3: An example of unconstrained DTW with two series from the GunPoint dataset [65]. DTW mitigates the misalignment of the series by matching the peaks and troughs of the data.

One of the first mainstream applications of DTW to time series data mining was reported in [62]. For TSC specifically, an early use of DTW was documented in [92]. The authors investigate the potential of DTW by first noting the improved performance over Euclidean distance with NN classifiers, but note the extra computational effort involved. This motivated an investigation that resulted in the introduction of the *R-K band* for DTW, a constraint for DTW that can be generalised to a warping of any arbitrary shape and size to accelerate computation and improve accuracy of DTW-based classification. They report results on three datasets to demonstrate the accelerated performance without detriment to classification decisions.

A further speed-up technique is proposed in [106], which involves reducing training set sizes to reduce the quantity of calculations, rather than improving the calculation itself. The authors initially claim that NN classification is very difficult to beat, and support this claim by providing an interesting comparison with other published methods to demonstrate where NN classifiers with DTW outperform more complex alternatives. However, it should be noted that while this provides interesting information, the results may include selection bias since they are optimistic cases selected to motivate their

approach. However, using this information they advocate using NN classifiers for TSC, and propose a speed-up for DTW that is based on numerosity reduction. They use an initial experiment to observe a relationship between dataset size and DTW warping constraints, where they observe that small warpings are favourable when many training instances are available. They configure their approach to initially limit warping in very large dataset, and then gradually increase the allowed warping as they remove instances from a dataset. They demonstrate that their approach is faster than using the standard implementation of DTW while still providing comparable results.

In [35], the authors carry out one of the largest investigations of NN classifiers with alternative similarity measures and representations. They include 8 time series representations and 9 similarity measures. Of note, these measures include the Euclidean distance, DTW, and edit distance-based measures including longest common subsequence (LCSS), edit distance on real sequence, and edit distance with real penalty (these similarity measures are discussed further in Section 2.3.6 and Section 2.3.7). Firstly, they note that as the quantity of training data increases, the accuracy of DTW and the edit distance-based measures converges with the Euclidean distance. This aligns with the original observation of [106], who favoured small warpings (since allowing no warping provides no elasticity, hence is equivalent to Euclidean distance) with large quantities of training data when investigating constraints with DTW. However, while they note that this observation is true for problems with abundant training data, elastic measures often outperform Euclidean distance when training data is limited. Their second key observation is that constraining measures such as DTW and LCSS reduces computation cost while giving equivalent or better classification accuracy. Thirdly, they do not find any conclusive evidence to suggest that any of the measures that they test outperform DTW. In fact, DTW outperforms some of the more recently proposed measures. Finally, they note that if a similarity measures does not provide adequate classification accuracy, the introduction of further training data often leads to improved accuracy. If such data is not available, they suggest that it may be beneficial to explore additional similarity measures that were not used in their work.

2.3.4 Derivative Dynamic Time Warping

Keogh and Pazzani [63] proposed a modification of DTW called Derivative Dynamic Time Warping (DDTW) that first transforms the series into a series of first order differences. The motivation for DDTW was to introduce a measure that avoids *singularities*, where a single point on one series may map onto a large subsection of another time series and

create pathological results. Given a series $\mathbf{a} = \{a_1, a_2, \ldots, a_m\}$, the difference series is $\mathbf{a}' = \{a'_1, a'_2, \ldots, a'_{m-1}\}$ where a'_i is defined as the average of the slopes between a_{i-1} and a_i, and a_i and a_{i+1}, i.e.

$$a'_i = \frac{(a_i - a_{i-1}) + (a_{i+1} - a_{i-1})/2}{2}, \tag{2.14}$$

for $1 < i < m$. DDTW is designed to mitigate against noise in the series that can adversely affect DTW, and has also been used in conjunction with standard DTW to simultaneously calculate similarity between series [46].

2.3.5 Weighted Dynamic Time Warping

A weighted form of DTW (WDTW) was proposed by Jeong *et al.* [58]. WDTW adds a multiplicative weight penalty based on the warping distance between points in the warping path. It favours reduced warping, and is a smooth alternative to the cut-off point approach of using a warping window. When creating the distance matrix M, a weight penalty $w_{|i-j|}$ for a warping distance of $|i - j|$ is applied, so that

$$M_{i,j} = w_{|i-j|}(a_i - b_j)^2. \tag{2.15}$$

A logistic weight function is proposed in [58], so that a warping of a places imposes a weighting of

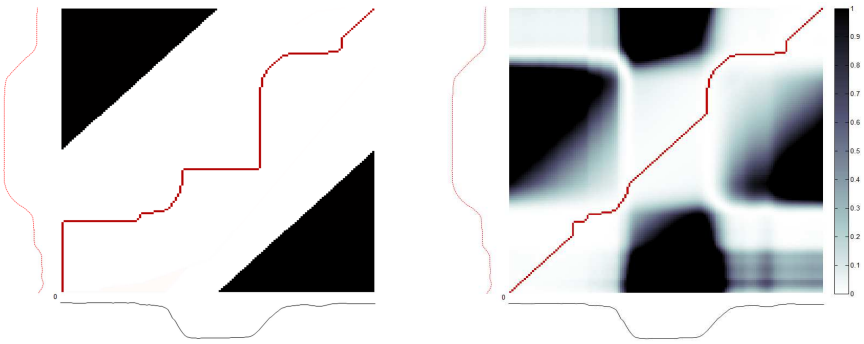
$$w(a) = \frac{w_{max}}{1 + e^{-g \cdot (a - m/2)}}, \tag{2.16}$$

where w_{max} is an upper bound on the weight (set to 1), m is the series length and g is a parameter that controls the penalty level for large warpings. The larger g is, the greater the penalty for warping.

2.3.6 Longest Common Subsequence Distance

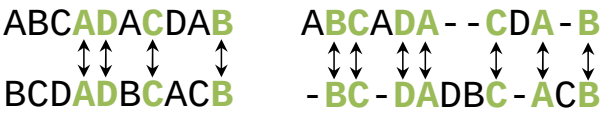
The Longest Common Subsequence (LCSS) distance is based on the solution to the longest common subsequence problem in pattern matching [53]. The typical problem is to find the longest subsequence that is common to two discrete series based on the edit distance. An example using strings is shown in Figure 2.5.

This approach can be extended to consider real-valued time series by using a distance threshold ϵ, which defines the maximum difference between a pair of values that is allowed for them to be considered a match. LCSS finds the optimal alignment between two series by inserting gaps to find the greatest number of matching pairs.



(a) (b)

Figure 2.4: Example distance matrices for the two GunPoint series from Figure 2.3 with warping paths for a 50% warping window with DTW (a) and WDTW with a penalty value of 0.1 (b). The dark areas in (a) depict the limits of the warping window where the path may not pass, whereas in (b) the dark areas represent areas that are highly weighted. It can be seen that the gradient of the weighting function encourages the path to avoid highly weighted areas in (b), but it does not strictly prevent traversal of those areas as a warping window does.



(a) (b)

Figure 2.5: An example of the LCSS problem. The example in (a) shows a pairwise matching of the two strings, while (b) demonstrates an alignment that allows shifting within the strings to allow for more matches to be made. The example in (a) is analogous with the LCSS distance with no elasticity, whereas (b) represents the LCSS distance with full elasticity. This can be controlled using a parameter akin to the warping window in DTW.

.

The LCSS between two series **a** and **b** can be found using Algorithm 1, and the LCSS distance between **a** and **b** is

$$d_{LCSS}(\mathbf{a}, \mathbf{b}) = 1 - \frac{LCSS(\mathbf{a}, \mathbf{b})}{m}. \tag{2.17}$$

Algorithm 1 LCSS $(\mathbf{a}, \mathbf{b})$

Let L be an $(m+1) \times (m+1)$ matrix initialised to zero.
for $i \leftarrow m$ to 1 **do**
  for $j \leftarrow m$ to 1 **do**
    $L_{i,j} \leftarrow L_{i+1,j+1}$
    if $a_i = b_j$ **then**
      $L_{i,j} \leftarrow L_{i,j} + 1$
    else if $L_{i,j+1} > L_{i,j}$ **then**
      $L_{i,j} \leftarrow L_{i,j+1}$
    else if $L_{i+1,j} > L_{i,j}$ **then**
      $L_{i,j} \leftarrow L_{i+1,j}$
return $L_{1,1}$

2.3.7 Edit Distance with Real Penalty

Other edit distance-based similarity measures have also been proposed. One such approach is edit distance on real sequences (EDR) [25]. Like LCSS, EDR uses a distance threshold to define when two elements of a series match, but also includes a constant penalty that is applied for non-matching elements and where gaps are inserted to create optimal alignments. However EDR does not satisfy the triangular inequality, as equality is relaxed by assuming elements are equal when the distance between them is less than or equal to ϵ. This was revised in [26], where edit distance with real penalty (ERP) was introduced. The motivation for ERP is that it is a metric as it satisfies the triangular inequality by using 'real penalty', which uses the distance between elements when there is no gap and a constant value g for when gaps occur. The ERP distance between element i of series a and element j of series b is

$$ERP(a_i, b_j) = \begin{cases} |a_i - b_j| & \text{if } |a_i - b_j| \leq \epsilon \\ |a_i - g| & \text{if } b_j \text{ is a gap} \\ |b_j - g| & \text{if } a_i \text{ is a gap,} \end{cases} \tag{2.18}$$

and the full ERP distance between series a of length m and series b of length n is given recursively as

$$d_{ERP}(a,b) = \begin{cases} \sum_{i=1}^{n} |b_i - g| & \text{if } m = 0 \\ \sum_{i=1}^{m} |a_i - g| & \text{if } n = 0 \\ min\{d_{ERP}(Tail(a), Tail(b)\} + ERP(a_1, b_1), & \text{otherwise} \\ \quad d_{ERP}(Tail(a), b) + ERP(a_1, gap), \\ \quad d_{ERP}(a, Tail(b)) + ERP(gap, b_1)\}, \end{cases} \tag{2.19}$$

where $Tail(a) = \{a_2, a_3, ..., a_m\}$.

2.3.8 Time-Warp Edit Distance

Introduced by Marteau [79], Time Warp Edit (TWE) distance is an elastic distance measure that, unlike DTW and LCSS, is also a metric. It encompasses characteristics from both LCSS and DTW as it allows warping in the time axis and combines the edit distance with *Lp-norms*. The warping, called *stiffness*, is controlled by a parameter ν. Unlike a warping window that constrains a DTW search, stiffness enforces a multiplicative penalty on the distance between matched points. Setting $\nu = 0$ results in no stiffness, or *null stiffness*, giving a distance measure equivalent to a full DTW search. Setting $\nu = \infty$ gives Euclidean distance. TWED redefines the *insert*, *remove* and *match* operations used in edit distance, in favour of *delete_a*, *delete_b* and *match*. The *delete_a* operation occurs when an element is removed from the first series to match the second, and *delete_b* occurs when an element of the second series is removed to match the first. An *Lp-norm* distance calculation is used when matches are found, and a constant penalty value λ is applied when sequences do not match. The formal definition TWED can be found in [79], and a dynamic programming implementation is given in Algorithm 2.

2.3.9 Move-Split-Merge

Move-Split-Merge (MSM) was introduced in [101]. The authors motivate the introduction of MSM as it satisfies a number of desirable traits that they set out to incorporate into a single similarity measure: it is robust to temporal misalignments; it is translation invariant; it has a competitive quadratic run-time with DTW; and it is a metric. MSM is conceptually similar to other edit distance-based approaches, where similarity is calculated by using a set of operations to transform a given series into a target series. Each operation has an associated cost, and three operations are defined for MSM: move, split, and merge. Move is synonymous with a substitute operation, where one value is replaced

Algorithm 2 TWE Distance($\mathbf{a}, \mathbf{b}, \lambda, \nu$)

Let D be an $m+1 \times n+1$ matrix initialised to zero.
$D(1,1) = 0$
$D(2,1) = {a_1}^2$
$D(1,2) = {b_1}^2$
for $i \leftarrow 3$ to $m+2$ **do**
  $D(i,1) = D(i-1,1) + (a_{i-2} - a_{i-1})^2$
for $j \leftarrow 3$ to $n+2$ **do**
  $D(1,i) = D(1,j-1) + (b_{j-2} - b_{j-1})^2$
for $i \leftarrow 2$ to $m+2$ **do**
  for $j \leftarrow 2$ to $n+2$ **do**
    if $i > 1$ and $j > 1$ **then**
      $dist1 = D(i-1,j-1) + \nu \times |i-j| \times 2 + (a_{i-1} - b_{j-1})^2 + (a_{i-2} - b_{j-2})^2$
    else
      $dist1 = D(i-1,j-1) + \nu \times |i-j| + (a_{i-1} - b_{j-1})^2$
    if $i > 1$ **then**
      $dist2 = D(i-1,j) + (a_{i-1} - a_{i-2})^2 + \lambda + \nu$
    else
      $dist2 = D(i-1,j) + {a_{i-1}}^2 + \lambda$
    if $j > 1$ **then**
      $dist3 = D(i,j-1) + (b_{j-1} - b_{j-2})^2 + \lambda + \nu$
    else
      $dist3 = D(i,j-1) + {b_{j-1}}^2 + \lambda$
    $D(i,j) = \min(dist1, dist2, dist3)$
return $D(m+1, n+1)$

by another. Split and merge differ from other approaches, as they attempt to add context to insertions and deletions. Stefan *et al.* state that cost of inserting and deleting values should depend on the value itself and adjacent values, rather than treating all insertions and deletions equally (for example, as in ERP). Therefore, the split operation is introduced to insert an identical copy of a value immediately after itself, and the merge operation is used to delete a value if it directly follows an identical value. The formal definition of MSM can be found in [101], and a dynamic programming implementation is given in Algorithm 3 using the cost function

$$C(a_i, a_{i-1}, b_j) = \begin{cases} c \textbf{ if } a_{i-1} \leq a_i \leq b_j \textbf{ or } a_{i-1} \geq a_i \geq b_j \\ c + min(|a_i - a_{i-1}|, |a_i - b_j|) \textbf{ otherwise.} \end{cases} \tag{2.20}$$

Algorithm 3 MSM Distance($\mathbf{a}, \mathbf{b}$)

Let D be an $m \times n$ matrix initialised to zero.
$D(1,1) = |a_1 - b_1|$
for $i \leftarrow 2$ to m **do**
  $D(i,1) = D(i-1,1) + C(a_i, a_{i-1}, b_1)$
for $i \leftarrow 2$ to n **do**
  $D(1,i) = D(1,i-1) + C(b_i, a_1, b + i - 1)$
for $i \leftarrow 2$ to m **do**
  for $j \leftarrow 2$ to n **do**
    $D(i,j) = \min(D(i-1,j-1) + |a_i - b_j|,$
    $D(i-1,j) + C(a_i, a_{i-1}, y_j),$
    $D(i,j-1) + C(y_j, x_i, x_{i-1}))$
return $D(m,n)$

2.4 Standard Classification Algorithms

While the most popular approach toward solving TSC problems is to use NN classifiers, it is also possible to apply generic classification algorithms directly to time series data. If we simply ignore that the structure and ordering of attributes is often an important feature of TSC problems, we can consider the attributes as independent readings and pose them as a standard classification problem. Though it may seem unintuitive to discard this information, it allows us to leverage from the wealth of algorithms proposed in the general classification literature. Furthermore, we can potentially transform time series to extract features (See Section 2.6 for more details), which would result in data that could be applied to general classification algorithms. In this section we describe

numerous classifiers that are used at various stages throughout this thesis.

2.4.1 Naïve Bayes

Naïve Bayes is a simple classification algorithm that assumes strong independence between attributes, hence the 'naïve' moniker. Though this may be overly simplistic for some problems, especially for time series where attributes are by definition strongly dependent, Naïve Bayes has been proven to be a fast and effective classifier in many problem domains, especially in text analysis [100] and spam filtering [97].

The implementation of Naïve Bayes classifiers is based on Bayes Theorem. Given a dataset of time series T with n series of length m, and the set of possible class values C, we must calculate the probability distribution for each attribute and class value, $p(T_i|c_k)$, and the probability of each class arising, $p(c_k)$. Given a query series $a =< a_1, 1_2, ..., a_m >$ with an unknown class, the class c^* is assigned by the classifier using the *maximum a posteriori* decision rule, which selects the class value with the highest estimated probability given the observed attributes. This is calculated as:

$$c^* = \max_{c_k \in C} p(c_k) \prod_{i=1}^{m} p(a_i|c_k) \tag{2.21}$$

This approach can be extended to consider dependencies between attributes. One of the key principles of a Naïve Bayes classifier is that attributes are assumed to be independent; by forming a directed graph between dependent attributes, probabilities for attributes represented at child nodes can be influenced by the outcome of parent attributes. This is known as a Bayesian Network.

2.4.2 C4.5 Decision Tree

The C4.5 (also known as J48) decision tree classifier was first introduced in [86] and is arguably the most popular decision tree implementation in the classification literature [66, 42, 24, 44, 87]. The C4.5 algorithm uses a greedy top-down approach for recursively building decision tree classifiers. The algorithm begins by observing whether the data consist of a single class; if this is true, the algorithm has met the stopping condition and a leaf node is created. If not, all attributes are evaluated to identify the most informative for splitting the data, and the data are partitioned according to the selected attribute. This process is then repeated on each subsequent data partition until every leaf node in the tree contains a single class value. A final step is then invoked to prune the tree by using the training data to observe whether removing nodes can lead

to increased performance in an attempt to avoid over-fitting.

A fundamental operation in building the decision tree is identifying the best attribute for partitioning data. The C4.5 algorithm uses the gain ratio for this purpose. Given a set of training time series T, each with m attributes $A = \{A_1, A_2, ..., A_m\}$, the gain ratio for the k^{th} attribute A_k is calculated as:

$$GainRatio(T, A_k) = \frac{InfoGain(T, A_k)}{H(T)}, \tag{2.22}$$

The calculation of the gain ratio requires two further equations: entropy and information gain. Entropy was introduced in [98] to measure the uncertainty of a random variable X that can take the values $X_v = x_1, x_2, ..., x_n a$, and is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i). \tag{2.23}$$

Information gain is the expected reduction in entropy due to splitting on a given criteria. The information gain for attribute A_k is defined as:

$$InfoGain(T, A_k) = H(T) - \sum_{v \in Values(A_k)} \frac{|T_v|}{|T|} H(T_v), \tag{2.24}$$

where $Values(A_k)$ is the set of all values that attribute A_k can take on, and $|T_v|$ is the cardinality of the set of readings in T that take on value v for attribute A_k. Using these definitions, the best attribute to split on A^* according to gain ratio can be found as:

$$A^* = \max_{A_k \in A} GainRatio(T, A_k) \tag{2.25}$$

The data is partitioned according to A^*, and the process is continued on each partition until a full tree is constructed.

2.4.3 Support Vector Machine

Support Vector Machines (SVM) were introduced in [28] and have been used extensively in the literature [84], with many applications with time series data for financial forecasting [83, 82, 103]. The simplest example of an SVM expects a problem to be linearly separable. For an example with a simple dataset T with two possible class labels

$C = \{1, -1\}$, the objective is to build the classifier $f(T)$ such that:

$$f(T) = \begin{cases} c^* = +1 & \text{if} \geq 0 \\ c^* = -1 & \text{if} < 0. \end{cases} \tag{2.26}$$

where $f(T)$ takes the form of:

$$f(T) = w \cdot T + b. \tag{2.27}$$

In this equation w is a weight vector normal to $f(T)$ and b is the bias that refers to the offset of $f(T)$ along w. The objective of the SVM classifier is to find the optimal hyperplane $f(T)$ to separate the classes in T. A simple method is to iteratively update w until all training instances are correctly classified (or it is found that it is not possible). More complex techniques include those that seek to find the maximum margin between classes, or use regularisation parameters for soft margins to avoid over-fitting through allowing a degree of misclassification during training. It is also common that problems are not linearly separable, so often non-linear kernel functions are used to transform data into a space where the problem is linearly separable. This is reflected in our experiments throughout this work, as linear SVM classifiers are often implemented alongside quadratic SVM classifiers.

2.4.4 Random Forest

Random forest [16] is an ensemble classification approach that uses many constituent decision tree classifiers. The goal of the random forest algorithm is to inject diversity into predictions through training constituent trees with random attribute subsets.

The forest contains k trees, where each tree is trained by initially being assigned a random subset S of the training data T. At each node in the tree, a random subset of b attributes is selected, and the best attribute in the sample is selected for partitioning the data. The procedure for building the constituent trees is highly related to how C4.5 classifiers are formed, with the main distinctions being the random attribute sampling rather than using the full data, and the measure of splitting quality; the random forest algorithm uses the GINI index as an alternative to information gain (previously introduced by the same author in [17]). The final classification prediction for a test instance is the modal prediction across all trees.

A key part of forming the classifier is to ensure that constituent trees are not highly-correlated; the greater the inter-tree dependence within the forest, the greater the error rate will be as diversity will be reduced. Therefore setting b is an important part of the

training phase of the algorithm, as lower values reduce the inter-dependence of trees, but also potentially lowers the performance of individual trees, so the optimal value for b should be found during training.

2.4.5 Rotation Forest

The rotation forest algorithm [94] is similar to random forest as it is also a tree-based ensemble approach. The classifier is created by dividing the training data into k subsets, where each is transformed using principal component analysis (PCA). PCA transforms the set of attributes in the data into an alternative set of uncorrelated variables, and the resulting principal components are used to train k C4.5 decision tree classifiers. The final prediction for a test instance is obtained in the same manner as the random forest classifier, where the majority class value from the constituent classifier predictions is selected as the output prediction.

2.5 Ensemble Classifiers

An ensemble of classifiers is a set of base classifiers, where individual decisions are combined to classify new examples through combining predictions into a single output. A key aim when building an ensemble classifier is to introduce new elements into the classification model to inject diversity. We have already considered two ensemble classifiers during the discussion of standard classifiers in Section 2.4: Random Forest and Rotation Forest.

Typical techniques for building ensemble classifiers include: creating a heterogeneous ensemble of different classification approaches; modifying the data used to train each constituent classifier, such as by resampling the data or replicating instances; selecting different sets of attributes to train each classifier on; and modifying each classifier internally by reweighting training data.

Two ensembling approaches in particular have been adopted frequently in the literature: bootstrap aggregation, or *bagging*, and *boosting*. In this section we introduce these approaches and discuss ensemble classifiers that have been used in the TSC literature. We then go on to define a simple heterogeneous ensemble that we use later in this thesis, formed using the standard classification algorithms in Section 2.4 as constituents, and demonstrate an example of building the ensemble in the time domain.

2.5.1 Bagging

Bootstrap aggregation, or *bagging*, is an ensembling approach that is designed to increase the stability in predictions by training constituent classifiers with different subsets of the training data, such as in the Random Forest and Rotation Forest algorithms that were introduced in Section 2.4. However, bagging specifically uses bootstrap samples [37], where samples from a dataset are drawn randomly with replacement. This effectively means that a single instance may appear in the training data for multiple constituents. The typical approach for building an ensemble classifier E with bagging is outlined in [15] and summarised in Algorithm 4, where C is the set of constituent classifiers, T is the training data, and b is the size of each training sample. Once the constituent classifiers are trained, the prediction for a test instance is produced by taking the majority class decision across all constituents.

Algorithm 4 Bagging(**C**, **T**, b)

1: $E = \emptyset$;
2: **for all** classifiers $\mathbf{C}_k$ in **C do**
3: $\quad T_k = bootstrapSample(T, b)$;
4: $\quad C_k$.buildClassifier(T_k);
5: $\quad E = E \cup C_k$;
6: **return** E;

2.5.2 Boosting

The key concept of boosting is to take a classifier that is considered a *weak-learner* and improve it through resampling the training data to weight against misclassified training instances in previous iterations. A *weak-learner* can be considered as any classifier that is a relatively poor solution to a problem, but is more accurate than random guessing. The first polynomial-time implementation of boosting was documented in [96], and the work in [40] extended this implementation to make the runtime more feasible. The most influential implementation of boosting was provided when the authors of the previous two works combined to create an adaptive boosting algorithm, or *AdaBoost* [41].

The general boosting algorithm uses an iterative training approach to assign weights to training instances. Given a classifier, the first training iteration will consider each instance with an equal weight, and a weighted accuracy is calculated. After the first iteration, the examples that are misclassified are given a greater weight, while those that were correctly classified are given a lower weight. In classifiers where weighting is

not possible, this can occur through replication of instances where a greater weight is desired. Then a new base classifier is trained using these updated weights with the goal of producing the lowest weighted accuracy. By reweighting training instances, subsequent classifiers are forced to focus on instances that were originally misclassified, creating a diverse pool of base classifiers. An ensemble can be formed using the base classifiers, and a test instances is classified according to the majority vote of the constituents.

2.5.3 Other Ensembles in the TSC Literature

Bagging and boosting are both popular approaches in the classification literature for building ensemble classifiers. However, novel ensembling approaches have also been proposed for TSC. For example, in [21], an ensemble algorithm is proposed using a regression model to create a fusion approach to combine classifiers built using various similarity measures. Support for this approach is provided through an experimental comparison over 35 datasets against DTW and SVM classifiers, suggesting that similarity in various problems can be better assessed using a fusion of measures, rather than only considering DTW.

In [34], the authors propose a tree-based ensemble classifier. The time series forest (TSF) uses an approach similar to random forest, as constituent trees are built through randomly selecting samples from the data using simple summary statistics, such as mean, slope, and variance. However, rather than using information gain or the GINI index to assess splitting criteria, TSF evaluates potential splits using a novel measure that combines entropy with a distance measure. The TSF approach is compared to DTW with 1-NN and the standard random forest algorithm over 45 datasets, and the results suggest that this approach significantly outperforms random forest.

2.5.4 A Simple Heterogeneous Ensemble

A key aim when building an ensemble classifier is to introduce new elements into the classification model to inject diversity. There are many approaches to this aim that have been documented in the literature, but one of the simplest techniques is to form a heterogeneous ensemble of different classification algorithms. A key aim in our work is to keep our methodology simple and transparent. Therefore we choose to use this approach for ensembling throughout our work, as not only does it allow us to create a diverse ensemble classifier, but the approach for building the ensemble is very transparent and conceptually simple.

The heterogeneous ensemble is formed by creating a pool of distinct classifiers where

there is no dependency between classifiers; for a given problem, each classifier is built in isolation and produces an individual prediction for a test instance. The ensemble uses a voting scheme to combine the predictions of each constituent classifier to output a final classification prediction. A pseudo-code implementation of a simple heterogeneous ensemble is defined in Algorithm 5, where $\mathbf{T}$ represents the training data, $\mathbf{C}$ is the set of classifiers in the ensemble, and q is the test series to be classified.

Algorithm 5 HeterogeneousEnsemble($\mathbf{C}$, $\mathbf{T}$, q)

```
1: predictions = ∅;
2: for all classifiers C_k in C do
3:     C_k.buildClassifier(T);
4:     predictions_k = C_k.classify(q);
5: return votingScheme.decide(predictions);
```

There are two main components of the heterogeneous ensemble that must be determined: the base set of classifiers, and the mechanism for combining individual predictions into a single output.

The Classifiers

The heterogeneous ensembles that we form throughout this thesis are composed of the standard classification algorithms surveyed in Section 2.4. These include: 1-*NN*, Naïve Bayes, Bayesian Network, C4.5 Decision Tree, Random Forest, Rotation Forest, Support Vector Machine with a linear kernel, and Support Vector Machine with a quadratic kernel. Each classifier is created using the default Weka [48] implementation.

Ensemble Voting Schemes

To maintain the transparency of the ensemble classifier we define, we initially propose three simple voting schemes for combining individual classifier predictions. The first scheme, *Equal*, places an equal vote on all constituent classifiers. After each classifier has made a prediction for a test series, the majority decision is selected as the prediction, and any ties are split randomly. The second voting scheme is *Best*. This approach is the opposite to *Equal* as only a single classifier is used for the final prediction and all other constituents are ignored. The *Best* constituent is determined as the classifier with the highest accuracy after carrying out a leave-one-out-cross-validation experiment on the training data, and ties are split randomly. The third strategy is *Proportional*. This approach combines characteristics of the previous approaches; all classifiers are used

when making the final prediction, but the votes are weighted proportionally for each classifier according to training accuracy. For example, if a classifier achieved 69% on the training data, it would have 0.69 of a vote in the final classification decision.

2.5.5 Heterogeneous Ensembles in the Time Domain

Using the specification that has been outlined in this section, we can create an example of a heterogeneous ensemble in the time domain. As previously discussed in Section 2.3, the current benchmark in TSC research is considered to be a 1-*NN* classifier using dynamic time warping with the warping window set through cross-validation (DTWCV). This gives us an opportunity to demonstrate a number of the key concepts introduced so far in this chapter; to motivate the use of ensembles in our work, we implement the standard classifiers discussed in Section 2.4 to form three heterogeneous ensembles in the time domain, using the three voting strategies that we outlined. We report test classification accuracies of the ensembles against DTWCV on the UCR datasets [65] (a commonly-used set of 46 TSC problems, which we outline in detail later in Section 3.1) in Table 2.1. To test for significance between these results, we can use a critical difference diagram as outlined in 2.2. The critical difference diagram summarising the data in Table 2.1 is shown in Figure 2.6.

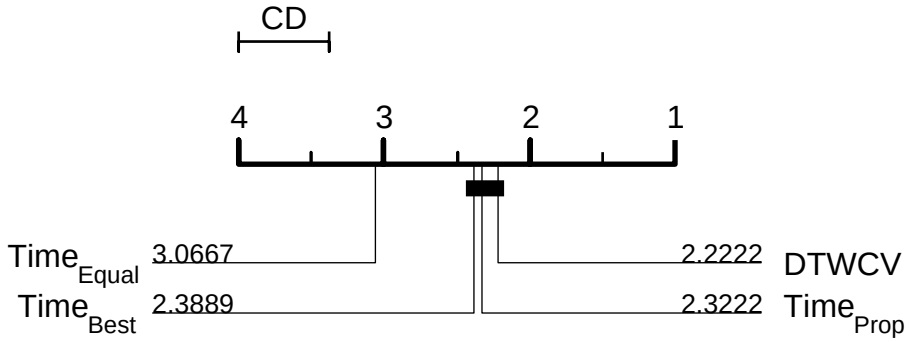


Figure 2.6: A critical difference diagram to compare DTWCV with three simple heterogeneous ensembles in the time domain.

.

The results demonstrate the *Proportional* and *Best* voting schemes for the heterogeneous ensemble are both significantly better than using the *Equal* weighting approach. However, though not significantly different, none of the results with the heterogeneous

ensemble in the time domain significantly outperform the benchmark of DTWCV. While this lends support to using ensembles for TSC, it does not advocate replacing DTWCV as the benchmark. This raises the question of whether there is an alternative ensembling approach for TSC in the time domain, such as investigating alternative elastic distance measures within an ensemble. This is investigated in the following section, and heterogeneous ensembles in other domains are investigated later in Chapter 7.

2.6 Time Series Transformations

An alternative approach to carrying out classification in the time domain is to consider alternative domains where discriminatory features may be more easily detected. Time series transformation algorithms process series to create alternative data representations. Most commonly in time series data mining (TSDM), the goal of such representations is to reduce time complexity when matching series, while minimising the loss of discriminating features. Such techniques are most commonly found in querying and indexing problems where transformations are applied to time series data to create a lower-dimension representation of the original data that can be used to approximate the Euclidean distance between series [35]. Such approaches can be extended to TSC to also accelerate run-time for very long series, and can also help mitigate noise in data. We provide a brief overview of time series transformation approaches that have been used in the literature that belong to two broad classes of transform: summary statistic transforms and compression-based transforms.

2.6.1 Summary Statistics and Feature Extraction

One of the simplest approaches for transforming time series is to extract summary features from the data, and then use the extracted features to train classifiers. We tested this approach in a case study published in [77] where the objective was to classify household electrical devices according to consumption readings recorded over 15 minute intervals. We considered daily and weekly data, resulting in instances of 96 and 672 in length respectively with 78,869 daily and and 9,215 weekly cases. Due to the very large volume of data, classification using a set of standard classifiers was very slow. To attempt to accelerate classification, we extracted 12 summary features from each series, including statistical summaries such as mean and standard deviation, and specialised summary information such as the first on time for a device, and the proportion of the day where the device was active (see Section 3.2.1 for more information on the data). This simple

Table 2.1: The results of the heterogeneous ensemble built in the time domain, used to create the critical difference diagram in Figure 2.6

Dataset	DTWCV	$\text{Time}_{\text{Best}}$	$\text{Time}_{\text{Equal}}$	$\text{Time}_{\text{Prop}}$
Adiac	61.13%	75.45%	70.33%	71.36%
Beef	66.67%	93.33%	80.00%	86.67%
Car	76.67%	81.67%	73.33%	80.00%
CBF	99.44%	90.22%	90.00%	90.56%
ChlorineConcentration	62.50%	81.41%	72.42%	72.89%
CinC_ECG_torso	92.90%	83.99%	78.33%	80.72%
Coffee	100%	100%	100%	100%
Cricket_X	75.38%	61.79%	57.44%	58.97%
Cricket_Y	79.49%	72.31%	71.28%	71.03%
Cricket_Z	82.31%	64.62%	63.85%	65.64%
DiatomSizeReduction	92.48%	95.10%	94.44%	95.10%
ECGFiveDays	80.02%	89.31%	88.73%	88.50%
FaceAll	80.77%	72.84%	76.21%	75.80%
FaceFour	89.77%	89.77%	89.77%	89.77%
FacesUCR	90.93%	78.05%	80.83%	80.98%
fiftywords	76.48%	63.08%	68.79%	69.45%
fish	83.43%	85.14%	83.43%	85.14%
GunPoint	91.33%	92.67%	89.33%	90.67%
Haptics	40.58%	44.81%	44.48%	45.78%
InlineSkate	38.55%	30.91%	33.82%	35.45%
ItalyPowerDemand	96.11%	96.89%	97.18%	97.18%
Lightning2	86.89%	72.13%	80.33%	78.69%
Lightning7	71.23%	71.23%	75.34%	73.97%
MALLAT	91.00%	91.00%	91.13%	90.96%
MedicalImages	73.95%	68.95%	72.89%	74.87%
MoteStrain	86.58%	90.42%	89.70%	90.26%
NonInvasiveFatalECG_Thorax2	86.77%	92.72%	91.60%	92.16%
OliveOil	86.67%	93.33%	90.00%	86.67%
OSULeaf	59.92%	54.13%	53.31%	57.02%
Plane	100%	97.14%	98.10%	98.10%
SonyAIBORobotSurface	69.88%	69.88%	75.04%	75.21%
SonyAIBORobotSurfaceII	85.73%	80.80%	80.27%	80.48%
StarLightCurves	90.30%	96.56%	93.55%	94.90%
SwedishLeaf	84.64%	86.08%	90.08%	90.24%
Symbols	93.07%	90.45%	90.05%	89.65%
SyntheticControl	98.33%	96.00%	96.33%	96.33%
Trace	99.00%	83.00%	81.00%	82.00%
TwoLeadECG	85.07%	84.64%	77.17%	79.89%
TwoPatterns	99.85%	90.68%	89.60%	91.73%
UWaveGestureLibrary_X	75.85%	75.01%	74.96%	75.85%
UWaveGestureLibrary_Y	68.65%	68.70%	68.09%	68.65%
UWaveGestureLibrary_Z	69.77%	70.49%	68.48%	69.77%
wafer	99.59%	99.38%	99.48%	99.51%
WordSynonyms	73.98%	61.76%	57.68%	59.25%
yoga	84.27%	80.77%	79.70%	81.53%

transformation approach reduced the length of daily cases by a factor of 8 and weekly cases by a factor of 56. We used this data to demonstrate that the performance of classifiers trained with the raw and transformed data did not provide significantly different results, while the classifiers built using transformed data were much faster.

Summary statistics have been used to great effect in other TSC work. For example, in [31], hand X-rays are classified by first being transformed into time series data, and then a simple summary filter is applied to extract features about the image that are used for classification. Also, the time series forest [34] algorithm introduced in Section 2.5 used statistics such as the mean, slope, and variance to build a tree-based ensemble classifier that significantly outperformed random forest. In [43], the authors propose a feature-extraction approach that is beneficial for very large data, extracting approximately 9,000 distinct features from data, including simple summary statistics, correlations and entropy calculations, with the aim of dimensionality reduction to avoid costly operations involved in computing distances between full series.

In addition to summary statistics, more complex features have also been extracted for TSC. For example, in [45] a technique is proposed to extract patterns from four TSC problems deriving features from series through resampling and interpolating data. They use this approach to search for local patterns within time series to build decision tree classifiers. This work is related to a recently proposed approach called time series shapelets [107]. A shapelet is a time series subsequence extracted from a dataset that is able to discriminates between classes based on local shape-based similarity. The authors define an algorithm to recursively extract the best shapelets from a dataset to create a decision tree classifier. Their motivation for using a tree-based approach is to highlight the intuitive nature of shapelets, while filling a niche in TSC literature for matching series according to local shape-based similarity. They demonstrate that their approach is effective for problems where time-domain approaches are not well suited, and demonstrate the explanatory power of shapelets through a number of case study examples. Shapelets are discussed further in Section 4.2.

2.6.2 Compression/Approximation-based Transforms

There are various compression-based transforms that have been applied to time series data to approximate similarity between series. One of the first applications of such an approach was in [38], where the authors used the Discrete Fourier Transform (DFT). DFT allows an input signal of finite length to be decomposed into a linear combination of sine and cosine waves, retaining only a subset of the resulting coefficients. This in

essence allows the time series data to be represented in the frequency domain, and the authors use the resulting coefficients to propose an approach for fast indexing of time series. The result of this transformation is primarily to reduce dimensionality of the data while attempting to retain the discriminatory information within the data. DFT and the related Fast Fourier Transform (FFT) have been used frequently in the TSDM literature in the application areas of classification, indexing, and querying.

A related approach is the Discrete Wavelet Transformation (DWT). Rather than using a combination of sinusoids to reconstruct time series, DWT uses wavelets in the transformation process. One of the first applications of DWT in TSDM was [23], where they created an indexing solution for time series by reducing dimensionality with DFT and Haar wavelets. Haar wavelets are popular due to the simple nature of the approach; DWT with Haar wavelets produces a transformed series by creating a lower-resolution version through averaging consecutive values, while retaining detail coefficients for reconstructing the original series [110].

In a similar vein to DFT and DWT, Singular Value Decomposition (SVD) has been applied to time series to reduce the dimensionality of time series for faster querying. In [67], this is achieved through representing series as linear combinations of eigenwaves that extracted from the raw data, and the dimensionality of the data is reduced through storing only a subset of the resulting principal components. They store only the most important components when transforming data, creating lossy representations that retain the majority of the explanatory power.

Piecewise Aggregate Approximation (PAA) is a compression-based transform that was first introduced in [60] that has many similarities to DWT with Haar wavelets. PAA is designed to be interpretable and conceptually simple by representing series as a combination of equal-length segments. The segments are computed by combining c successive readings and recording the average, where c is the compression value of the transform. Therefore if the resampling rate is a power of two, the PAA of a series is equivalent to the Haar DWT representation, as noted in [7]. A PAA-based approach for TSC is used in [45] where the authors propose a technique for extracting patterns from datasets on four TSC problems. The authors investigate using naive features derived from simply resampling and interpolating the data, and then build on this by searching for local patterns that appear within the approximated series to build decision tree classifiers. PAA has also been extended to Adaptive Piecewise Constant Approximation (APCA), where series are transformed using segments of varying lengths, rather than concatenating segments of equal length. The motivation for this adaptation of the PAA approach is to avoid removing maxima and minima in the compressed representation, as

the flexibility of using segments of variable length allows these to be incorporated in the transformed series.

A recently proposed representation related to PAA and APCA is Symbolic Aggregate Approximation (SAX) [71]. The transform is designed to leverage from the field of text mining, where many indexing and querying techniques have been highly developed for discrete data. The authors note that this idea is not novel itself, with previous methods existing to transform time series data into discrete series, SAX is the first symbolic representation that allows for dimensionality reduction, lower bounding, and transformation of streamed data. SAX represents time series as a string of characters from an alphabet α. The SAX transformation is achieved by firstly transforming data using PAA, and then using a set of breakpoints calculated from a Gaussian curve to assign characters from α to each piecewise segment. They then define a simple distance measure between SAX-transformed series that is based on the Euclidean distance, and demonstrate that the representation can be used in nearest neighbour and decision tree classifiers to accelerate performance without detriment to accuracy. The most prominent use of SAX in the TSC literature is in [72] where the authors propose a bag-of-patterns approach. This technique transforms series using SAX, and then passes a sliding window approach to extract *words* from the transformed series. A classifier is then built using frequency occurrences of words within series as features.

2.6.3 Transformation into Alternative Data Spaces

The majority of TSDM research using transformation approaches, such as DFT and DWT, use transformed series for faster approximation of the Euclidean distance between series in the time domain. Subsequently, the majority of research has focused on indexing and querying applications, rather than classification, and TSC research using alternative data representations has been minimal. Where transformation is used in the TSC literature, it is often embedded within the classifier (such as in [72]).

However, [3] proposes transforming time series into alternative data spaces to uncover similarity in other domains, rather than simply approximating Euclidean distance in the time domain. This is a key motivation for the thesis outlined in Chapter 1; we wish to search for shape-based and change-based similarity to uncover discriminatory features in datasets that are not suited to the time domain, and create a mechanism to decide which representation is best suited to a problem. We use the Power Spectrum (PS) to search for global shape-based similarity, shapelets for measuring local shape-based similarity, and the Autocorrelation Function (ACF) for assessing change-based similarity

between series. We discuss each of these approaches in detail in Chapter 4 and provide motivational examples using new datasets that were introduced into the literature as part of this thesis.

Chapter 3

Data

The aim of this work is to provide evidence in support of our thesis that the best way to approach TSC problems without prior knowledge is to first transform data into alternative representations, then build classifiers on the transformed data where discriminatory features may be more easily detected. To test this it is essential to have a wide and varied pool of problems to evaluate our proposed solutions with. Throughout this thesis, we evaluate our algorithms with the largest and most diverse collection of TSC problems ever assembled. Though the datasets used in each chapter vary slightly due to the availability of problems at the time of development, the final algorithm that we propose is evaluated on 72 datasets. To ensure that there is no selection bias in our problems, a large proportion of these datasets are made up of the 46 UCR time series datasets, a widely-used repository of problems that allow us to directly compare our results to other work. We also collect problems from other domains and introduce numerous datasets into the TSC literature. Such datasets include problems derived from household electricity consumption data, image outline problems taken from the MPEG 7 data, bone outline data derived from X-rays, and biology problems taken from motion capture of *Caenorhabditis elegans*. This chapter contains an overview of the nature of the datasets that we use, their origins, and explanations of how they were formed.

3.1 UCR Time Series Data Repository

The University of California: Riverside (UCR) time series data repository [65] is a publicly available store of TSC problems. First arising in 2003, the UCR repository has grown to include 46 problems, each split into training and testing sets. The procedure of explicitly splitting data into training and test sets is important to note, as this is a

Table 3.1: The 46 UCR datasets, divided into categories of problem type to aid interpretation.

Human Sensor Problems		
CinC_ECG_torso	ECGFiveDays	NonInvasiveFatalECG_Thorax1
NonInvasiveFatalECG_Thorax2	TwoLeadECG	
Image Outline Problems		
Adiac	DiatomSizeReduction	FaceAll
FaceFour	FacesUCR	fiftywords
fish	MedicalImages	OSULeaf
SwedishLeaf	Symbols	WordSynonyms
	yoga	
Motion Problems		
Cricket_X	Cricket_Y	Cricket_Z
GunPoint	Haptics	InlineSkate
UWaveGestureLibrary_X	UWaveGestureLibrary_Y	UWaveGestureLibrary_Z
Sensor Problems		
Beef	Car	ChlorineConcentration
Coffee	ItalyPowerDemand	Lightning2
Lightning7	MoteStrain	OliveOil
Plane	SonyAIBORobotSurface	SonyAIBORobotSurfaceII
StarLightCurves	wafer	
Simulated Problems		
CBF	MALLAT	SyntheticControl
Trace	TwoPatterns	

trend that is present in much of the TSC literature. Therefore to ensure our results remain consistent, our experiments are carried out by training classifiers exclusively on training data and reporting results on the test data only. When introducing our own TSC problems, we also adopt this convention and divide data into explicit training and test sets.

The full list of the 46 UCR datasets is presented in Table 3.1. The datasets are split by problem type; this informal grouping helps aid interpretation of the types of data that form the UCR repository, and allows us to make a more informed analysis of results. However, this is for information only and does not affect how we apply classifiers to problems of different types.

3.2 Electricity Consumption Problems

The first problems that we introduce are derived from electricity consumption data. Specifically, the datasets that we present are based on household device consumption and arise from two different sources. The first datasets were recorded during a prototype study by Green Energy Options (GEO), a company who specialise in creating smart energy metering devices. The second data source is a study commissioned by the government of the United Kingdom to measure many aspects within residential households,

including individual device consumption.

3.2.1 Visual Energy Trail (VET) Data

The Visual Energy Trial (VET) consisted of measuring power consumption for a variety of user-labelled devices within 187 households across the United Kingdom. To create the TSC problem, we extracted consumption data from the ten most commonly observed devices. These included: immersion heater, washing machine, fridge, freezer, fridge/freezer, oven/cooker, computer, television, and dishwasher. All monitoring devices sampled data in 25 minute intervals, from which we derived two classification problems. The first consisted of readings over a 24 hour period (96 attributes), and the second increased the observation window to a full week (672 attributes). After data cleansing and validation, the daily problem had 78,869 cases and the weekly problem had 9,215 cases. To better motivate the problem, Table 3.2 demonstrates numerous summary statistics recorded over the daily dataset.

Table 3.2: Summary statistics for the daily data set. Each data is averaged over all cases of the given class. For example, the minimum power usage in any one 15 minute period for a computer is 26.35 Wh when we average across all computers, assuming any power is being used at all. First usage represents the average attribute index where non-zero consumption is first observed in a day, and % on reflects the average proportion of the day that the device shows non-zero consumption.

	Computer	Dish Washer	Freezer	Fridge	Fridge/ Freezer	Immersion Heater	Kettle	Oven/ Cooker	TV	Washing Machine
Summary statistics for power usage when a device is in use (Wh)										
min	26.35	276.01	17.61	12.34	19.07	151.07	61.44	252.05	29.65	274.96
max	39.79	457.92	37.34	27.07	45.42	245.49	113.98	423.94	45.24	375.88
mean	33.13	365.13	27.14	20.10	28.91	201.80	84.94	328.79	39.36	324.14
std dev	6.42	102.69	8.11	4.55	7.28	75.65	27.10	114.63	11.67	81.13
skewness	0.07	0.03	-0.22	-0.55	0.45	0.11	0.17	0.26	-1.08	0.03
kurtosis	2.62	-1.44	1.35	0.64	5.05	0.53	-0.94	-0.73	3.42	-1.04
Summary stats of device usage tendencies										
% on	40	4	47	39	45	20	5	6	25	3
first usage	33.73	51.57	1.69	1.83	5.87	28.88	32.34	61.15	45.09	45.78
Summary stats of the number of time steps a device is on for										
num runs	3.03	2.13	23.18	17.91	14.27	6.55	4.55	1.94	2.71	1.70
run min	28.00	1.61	5.99	2.30	5.86	3.58	1.02	2.45	8.04	1.32
run max	33.68	2.17	9.60	5.90	13.17	7.99	1.27	3.36	16.80	1.59
run mean	30.54	1.88	6.97	3.55	8.73	5.43	1.07	2.84	11.82	1.44

The VET classification problems have several confounding factors that will make classification difficult. Firstly, the fact that measurements are summed over 15 minutes makes it harder to detect devices that peak over a short period. For example, a kettle will consume a large amount of power whilst on, but will only be on for two or three minutes; when summed over 15 minutes it will be harder to distinguish from a device

such as a dishwasher or washing machine, as these appliances consume lower power but will be on for the whole period. Secondly, there will be seasonal variation within the data, especially for devices such as immersion heaters. Thirdly, we would also expect it to be hard to distinguish between similar devices such as a fridge and a fridge/freezer and finally, we would expect considerable variation between different devices of the same class.

The objective of these problems is to identify the class of a device for a new user without any previous labelled consumption. Therefore we design the train/test split of the data to best avoid introducing bias into classification results. We believe that it will be much easier to identify a device for a single household, rather than across all households, so keep data from a specific household exclusive to either the training data, or the testing data. We originally introduced this problem in [77], and concluded that it was very difficult to distinguish devices that were of a very similar nature over 15 minute intervals. For example, refrigerators, freezers, and fridge/freezers all appeared very similar in the data, as did computers and televisions. Therefore in the final problems that we use in that work, we group those devices into two new classes: cold and screen. Figure 3.1 demonstrates example series from each of the classes.

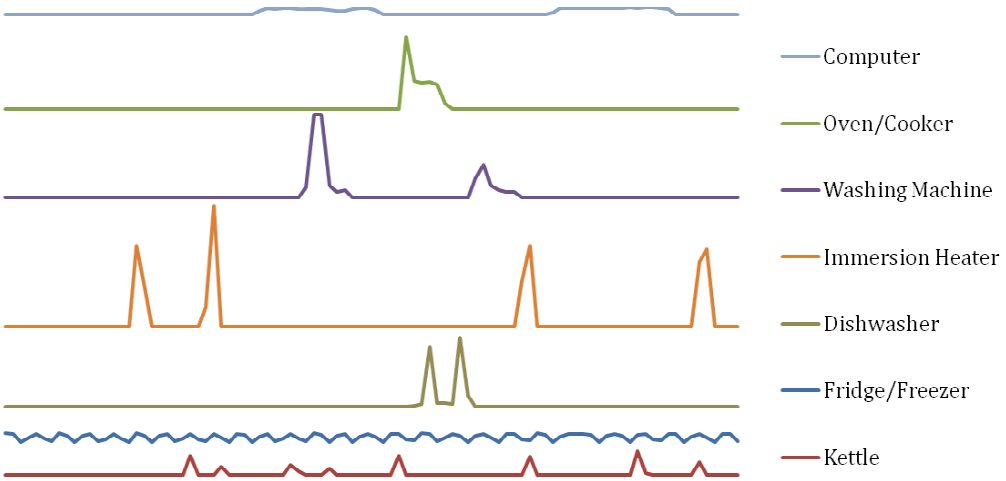


Figure 3.1: Example series taken from the VET data. The series shown are randomly selected cases from each class, sampled at 15 minute intervals over a single day.

3.2.2 Household Energy Study (HES) Data

The second set of household device classification problems that we introduce in this thesis are derived from data recorded as part of government sponsored study [104]. The intention was to collect behavioural data about how consumers use electricity within the home to help reduce the UK's carbon footprint. The aim is to reduce CO_2 emissions 80% by 2050, and it is clear that reductions in household consumption will be crucial to meet this goal. Much recent attention has been placed on how to reduce consumer consumption without a perceived reduction to quality of life, and one attempt to do this by the UK government is to install smart meters with real-time displays into every household. At a cost of £11.1bn, the project aims to alert consumers of their current consumption with the primary objective to show consumers their real-time spending, with the hope that this insight will cause them to reduce unnecessary consumption to save money. However, this project will also create vast streams of data from the 30 million households that will be monitored, so it is imperative that further information is extracted from the data to support other projects, such as creating smart energy grids and managing the creation of renewable energy.

We provide five new classification problems that focus on domestic appliance classification from electricity consumption. The datasets that we introduce support the goal of promoting further understanding of domestic electricity consumption by classifying appliances according to their electricity usage profiles. The data contains readings from 251 households, sampled in two-minute intervals over a month. When designing the classification problems we took our previous work in creating the VET classification problems (Section 3.2.1) into account, influencing the creation of these datasets in two regards. Firstly, we found that if consumption data from a specific device in a given household were included in both the training and testing data, bias was introduced into classification results as decisions were made by matching the specific device, rather than classifying the class of the devices. Secondly, we noted it was difficult to differentiate between devices with similar purposes and behavioural patterns. We use this previous insight to create two distinct types of problem: problems with similar usage patterns (Refrigeration, Computers, Screen) and problems with dissimilar usage patterns (Small Kitchen and Large Kitchen). The aim is that problems with dissimilar usage patterns should be well suited to time-domain classification, whilst those with similar consumption patterns should be much harder. The five problems we form are summarised in Table 3.3. These datasets were introduced into the literature in [75].

Table 3.3: The five new TSC problems with class values

Problem	Class Labels
Small Kitchen	Kettle, Microwave, Toaster
Large Kitchen	Dishwasher, Tumble Dryer, Washing Machine
Refrigeration	Fridge/Freezer, Refrigerator, Upright Freezer
Computers	Desktop, Laptop
Screen	CRT TV, LCD TV, Computer Monitor

3.3 Hand Outline Datasets

We provide numerous classification problems derived from hand X-rays, where the data was originally obtained from [85]. The data consists of X-rays of the hands of juvenile participants, with association information about the age of the subject. The data contains many challenging aspects for analysis; all were recorded as part of routine medical care as it would be unethical to subject participants to unnecessary radiation from the imaging procedure, so no effort is made to ensure that the images are consistent. They were each initially recorded simply for use with that participant, so often contain inconsistent lighting, differing hand posing, visible markers within the images, and so on. Therefore the process of extracting information was non-trivial, and formed a large component on a project to create a system for automated bone age assessment [30]. The objectives of this project lead to various datasets being produced, from which we derived 18 classification problems that we have used at various stages throughout this work. The first iteration of 8 datasets consisted on classifying bone age of participants from observing the outlines of 8 different phalanges in the hand. These datasets later evolved to create 10 more diverse problems, which were composed of problems determining whether image outlines were correct, the bone age of participants, and the estimation of the Tanner-Whitehouse score. Below we briefly explain the processing techniques that were used to create the classification problems from the original data, and provide an overview of the datasets that we have used in this thesis.

3.3.1 Data Preparation

The first stage in creating the TSC problems was to convert the 2-dimensional X-ray images into 1-dimensional time series. The outlines were automatically applied to over 1,300 images using the algorithm described in [32], and the results of the process were manually labelled by three human evaluators to judge whether the outlines were correct or incorrect. After this stage, 1,045 cases remained with outlines that were deemed to be accurate.

The next stage in the process was to isolate the individual bones from the outlines

that are important in the assessment of bone age. The bones that were extracted were the phalanges of the thumb, middle finger, and little finger. There are 3 phalanges in the little and middle fingers (distil, middle, and proximal) and two in the thumb (which does not contain the middle phalanx). For each correctly outlined X-ray, these were extracted by using the algorithm described in [31] to identify the tips and webs of the hand. From these positions, the axes of the thumb, middle, and little fingers were calculated by finding the mid-point between adjacent webs in the hand. In the cases of the thumb and little finger, where there is only one adjacent web for each, the axes were approximated by extending the line from the previous finger and calculating where this intersects the hand outline. Once provided with these axes, region-of-interest boxes were calculated for each of the eight bones as shown in Figure 3.2.

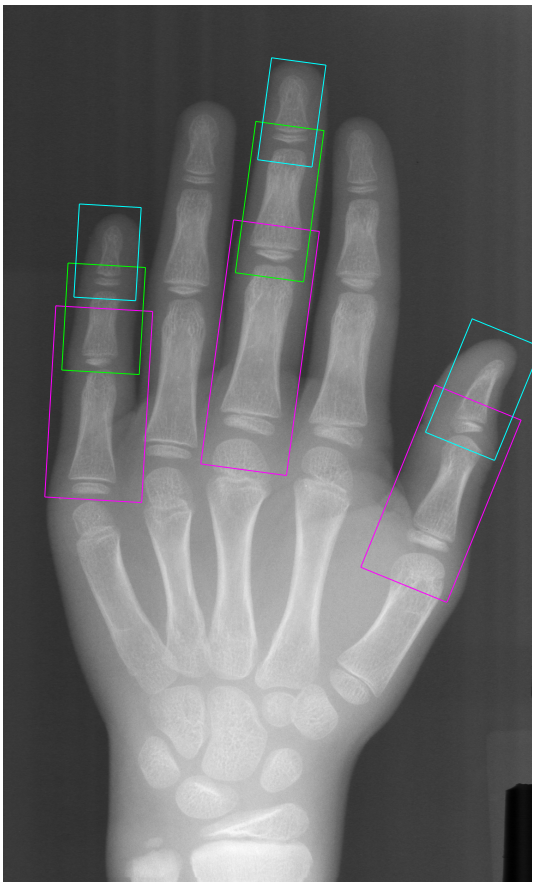


Figure 3.2: A hand X-ray with the eight bones boxed: proximal (purple, bottom), middle (green, middle), and distal (blue, top) phalanges. Note, the thumb does not have a middle phalange.

Each box was warped into a rectangular-shaped mesh (500 × 150 pixels) using a piecewise affine warp to create a new image for each of the 8 bones. Each image was converted into 1-d series, creating 1045 instances of data for each of the 8 bone types. A common method for converting an image into a 1-d series is to calculate a histogram of the image [81]. However, this approach doesn't incorporate any location information from the original image. To preserve this contextual information, the images were converted to 1-d series by resizing them to 30 × 9 pixels, each represented as a vector of length 270. This approach retains location information in the data while converting the images into 1-d series that can be posed as TSC problems.

The data generated throughout the various processing stages were used to form many TSC problems, which were created in two distinct iterations:

Phalanges

The first generation of the data provided 8 TSC problems, one for each of the extracted phalanges (*DP_Little*, *DP_Middle*, *DP_Thumb*; *MP_Little*, *MP_Middle*; *PP_Little*, *PP_Middle*, *PP_Thumb*). Each problem contained a total of 1,045 cases, which were split into 200 training instances 845 test instances. The problem was to classify the age group of subjects into classes of either 0-6, 7-12, or 13-19 years old.

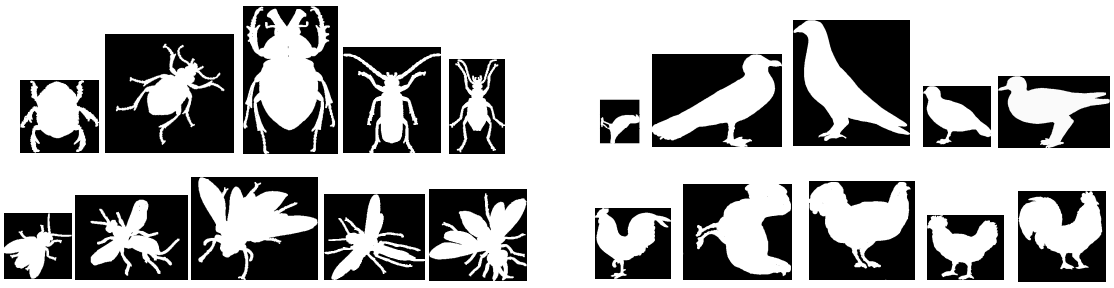
Outlines, Age, Tanner-Whitehouse

The second generation of data that we used mirrored the objectives of [30] much more closely. 10 problems were created in total, where the first 3 were to classify whether extracted phalanx outlines were correct for the three different types (*DistPhalanxOutline*, *MidPhalanxOutline*, *ProxPhalanxOutline*). The next three datasets were amalgamations of those used in the first generation datasets, where the problems were combined to create one dataset for each phalanx type with concatenated readings across the three digits for each subject, rather than an individual problem for each digit/phalanx combination (*DistPhalanxAge*, *MidPhalanxAge*, *ProxPhalanxAge*). Additionally, since these three problems are all aligned by subject, a problem was also created by concatenating the outlines of all 8 phalanges for each subject (*Phalanges*). Finally, the last three datasets represent the final aims in [30]. The main objective of the project was to automate the process of bone age assessment, where common practice involves an expert manually scoring X-rays of a subject to estimate the Tanner-Whitehouse score [102]. Therefore the final three problems take each of the phalanges and poses the problem of classifying cases by Tanner-Whitehouse score (*DistPhalanxTW*, *MidPhalanxTW*, *ProxPhalanxTW*).

3.4 MPEG-7 Problems

The MPEG-7 CE Shape-1 Part B [13] database is a freely available collection of binary images that were collated for testing MPEG-7 contour/image and skeleton-based descriptors [69]. The data contains 1,200 images in total, with 20 instances of 60 distinct classes. The classes vary in complexity and shape, with many classes of a similar nature to one another, or with discriminatory features embedded internally within the limits of the image.

We have created two problems using pairs of related classes: *Beetle/Fly* and *Bird/Chicken*. These problems were designed to be suitably challenging by including images that are not rotationally aligned and are of a visually similar nature, while still allowing for simple processing to transform into 1-d series. This is due to the make-up of the four classes that we consider; rather than containing intricate internal details or patterns, these classes consist of solid shapes. To demonstrate this, example series from the classes of the two problems are shown in Figure 3.3.



(a) (b)

Figure 3.3: Example images from the (A) *Beetle/Fly* and (B) *Bird/Chicken* TSC problems extracted from the MPEG-7 data.

Due to the nature of classes that we select, the process that we use to transform the images into TSC problems does not need to consider the internal details of the images (unlike the data previously described in Section 3.3). This simplification of the problem enables us to obtain 2-d landmark data of the image outlines by using a simple thresholding technique (such as that described in [81]). Once the outlines are identified, we resampled the coordinates for each image outline to a consistent length of 512 readings. The 2-d landmarks were then converted into 1-d series by calculating the distance between each point and the center of the image. An overview of this process is presented in Figure 3.4. Once all of the series were transformed using this approach, the two TSC

problems *Bird/Chicken* and *Beetle/Fly* were formed by assigning half of the series to training data, and half to testing data.

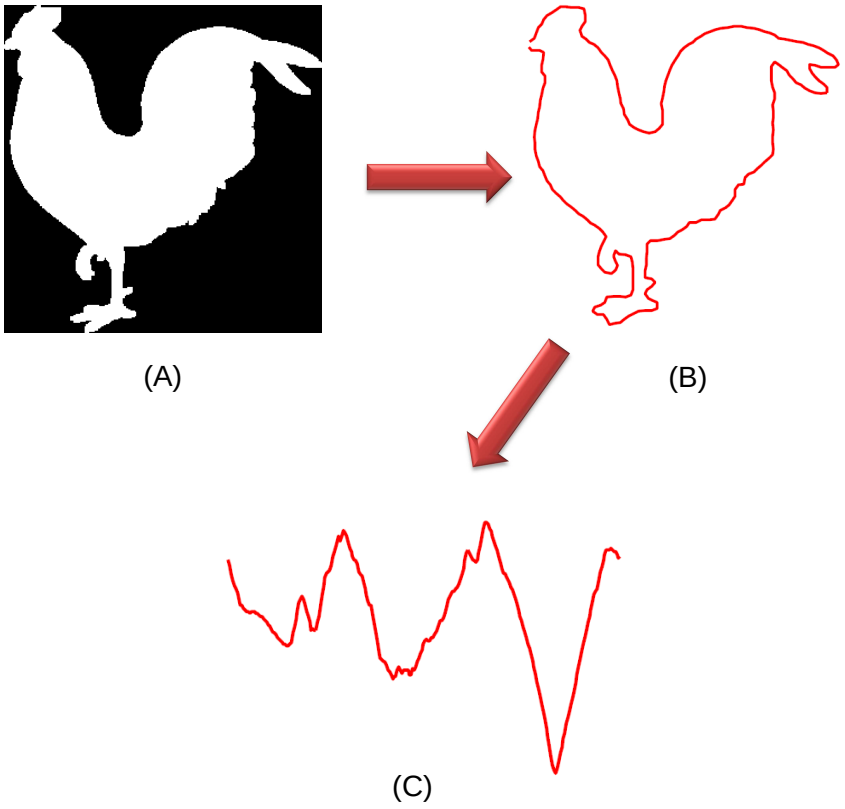


Figure 3.4: An overview of the data preparation for the MPEG-7 TSC problems. (A) shows an original image, and (B) represents the outline after a simple thresholding approach has been applied. The 1-d time series (C) is created by resampling the outline to 512 readings, and calculating the Euclidean distance from the centre of the image to each point.

3.5 Caenorhabditis elegans

Caenorhabditis elegans is a type of roundworm that is commonly used as a model organism in the study of genetics. The movement of these worms is known to be a useful indicator in the understanding of behavioural genetics. Brown *et al.* [18] describe a system for recording the motion of worms on an agar plate and measuring a range of human-

defined features [109], and it has been demonstrated that the shapes that *Caenorhabditis elegans* adopt during movement on an agar plate can be represented by a combination of four base-shapes, which are called *eigenworms*.

Once the outline of a worm is extracted, each frame of its motion can be described by four scalars that represent the amplitudes along each dimension when the shape is projected onto the set of four eigenworms (as shown in 3.5, originally taken from [18]). The resulting data that we use contains 257 cases of worm motion, consisting of five types of worm. The first is a control group (N2 reference strain, 109 cases), and four mutant types: goa-1 (44 cases), unc-1 (35 cases), unc-38 (45 cases), and unc-63 (25 cases). The data are down-sampled to second-long intervals, resulting in instances with 900 observations.

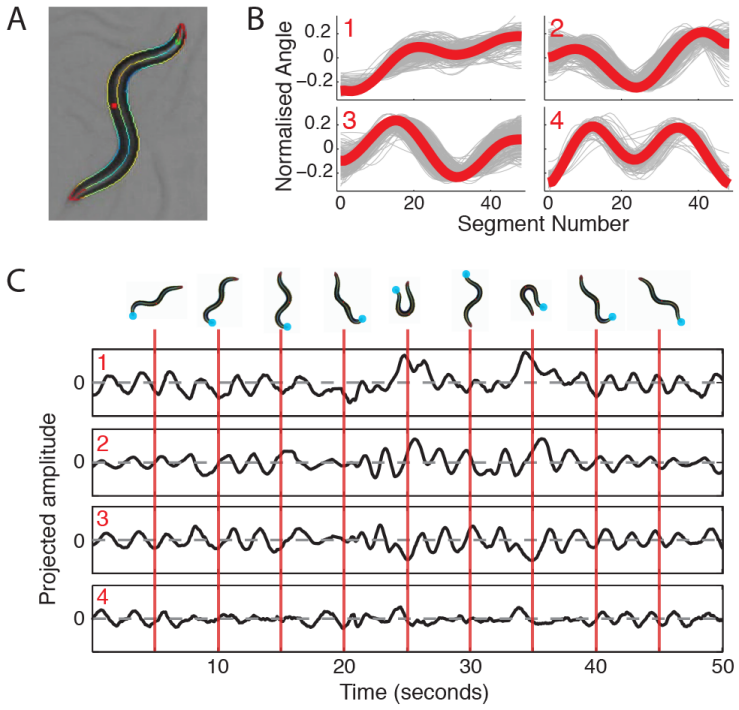


Figure 3.5: An figure originally taken from [18] to demonstrate how the datasets were created. A worm on an agar plate (A) is shown, alongside the four representative eigenworms (B). Example series generated from the motion data are shown in (C)

We use this data to pose two classification problems. The objective of the first problem is to classify individual worms as arising from either the control group, or being a mutant type. The four mutant types are combined into a single class, resulting in a

binary classification problem *Worms2*. The second TSC problem that we create is a specialisation of the first; the objective is to classify worms as part of either the control group, or a specific mutant strain. This problem contains 5 class values, including the N2 control strain and each of the five mutant types, and is called *Worms5*. As per the UCR repository [65] and the other datasets that we propose in this thesis, the data are split into training and testing partitions. We do this by randomly assigning 70% of the data to the training set and the remaining 30% to the test set. It should be noted that the instances are consistent between *Worms2* and *Worms5*, so data that appear in the training data for one problem will also be in the training data for the second problem.

Chapter 4

Time Series Similarity with Alternative Representations

Chapter 2 introduced the background for this thesis, highlighting numerous methods for transforming time series. Such approaches included representing time series with lower dimensionality, such as piecewise approximations and wavelet approaches, and also spectral approaches that have been used to represent data in the frequency domain for faster approximation of the Euclidean distance. However, the typical role of transformations in time series data mining has predominately been to increase the efficiency of algorithms by speeding up calculations, often for querying and indexing applications, rather than searching for similarity in transformed spaces other than the time domain. A key component of our thesis stated in Chapter 1 is that through transforming TSC problems into other domains, discriminatory features can be discovered more easily.

In this section we explore three types of similarity that we wish to incorporate in our work:

- Similarity in global-shape;
- Similarity in local-shape;
- Similarity in change.

We outline the techniques that we utilise in pursuit of representing data for exploring these three types of similarity, and include motivational examples of each using selected datasets described in Chapter 3.

4.1 Global Similarity in Shape: Power Spectrum

For searching for discriminatory features in global-shape, we use the power spectrum (PS) to represent time series in the frequency domain. The PS for a series $T =< t_1, t_2, ..., t_m >$ is calculated by first transforming T using the discrete Fourier transform (DFT), which represents the data as a linear combination of sines and cosines to describe how much information is stored in the series at different frequencies. The transformed series T is represented as:

$$f(T) = \sum_{i=1}^{m} a_i \cos(2\pi w_i x) + b_i \times \sin(2\pi w_i x), \tag{4.1}$$

with amplitudes a and b, and phase w. The transformed series T_F is typically then expressed as a series of pairs for each frequency,

$$T_F =< (a_1, b_1), (a_2, b_2), ..., (a_m, b_m) > . \tag{4.2}$$

In previous TSDM work, DFT is often computed using the Fast Fourier Transform (FFT) to approximate the similarity between series in the time domain with a lower dimensionality. This is achieved by truncating the data and calculating the Euclidean distance between Fourier-transformed series. However, we wish to use the transform to represent series in the frequency domain. To observe similarity in the global shape of the series, we are interested in whether a sinusoid is present, regardless of phase. Therefore we use the PS to describe the power at each frequency, where the periodogram of the series is formed by squaring and adding the Fourier coefficients. We calculate the PS of the series T as:

$$T_P =< p_1, p_2, ..., p_m > \tag{4.3}$$

where

$$p_i = \sqrt{a_i^2 + b_i^2}. \tag{4.4}$$

We can simplify the resulting periodogram in two ways. Firstly, when working with zero-normalised data, the first Fourier coefficient will always be 0 since the mean of the series is 0. Therefore we can remove the first term without losing any information from the transformed data. Secondly, since DFT is periodic, the second half of the periodogram will be reflected, and we can discard the last half of the data. This gives us the final series:

$$T_P =< p_2, p_2, ..., p_{m/2} > \tag{4.5}$$

4.1.1 Motivational Example: Electrical Devices

An example of transforming three series from the Electrical Devices problem introduced in Section 3.2 is presented in Figure 4.1. The figure contains examples from three classes in the time domain: kettle (A), refrigerator (B), and television (C), and the corresponding PS-transformed series in (D), (E), and (F). The time domain example of the kettle shows two occasions of consumption at approximately 06:30 and 17:30. These readings are typical for a kettle, with very high consumption appearing in small bursts early in the morning and during the evening, likely corresponding to breakfast and dinner for the user. Due to the absence of consumption throughout the rest of the day, there are no frequencies in (D) that contain a relatively large amount of information. In contrast however, the time domain data of the refrigerator in (B) is very periodic, with a repeating motif of consistent consumption throughout the day. This is reflected in the PS in (E), where the information is contained across a small group of frequencies, with one in particular containing the majority of the information. The series of television consumption in (C) provides an example between these two extremes, with two bursts of consumption during the day in the raw data. The transformed series is shown in (F), where the power is more evenly distributed across a small group of frequencies.

4.2 Local Similarity in Shape: Shapelets

The previous section introduced an approach for transforming time series into the frequency domain for uncovering global similarity in shape. However, shape-based similarity is not always global. For example, consider an electrocardiogram (ECG) heartbeat for a patient where a single beat arrhythmia is indicative of a heart condition. If this were captured as a time series and compared to a series of normal ECG behaviour, it would be very difficult to detect a difference due to the presence of many regular heartbeats. The discriminatory feature in this case would be described by the presence of a small local shape within the series indicating an irregular beat, which would likely be missed in the frequency and time domains as the structure and global shape of the data would still be very similar. We consider extracting time series shapelets for detecting local shape-based similarity between series.

Shapelets are a time series data mining primitive that can be used to determine similarity based on small common shapes occurring at any point in a series [107]. As introduced in Section 2.6.2, the process for shapelet extraction is embedded within a decision tree classifier. While the original work does not describe a data transformation

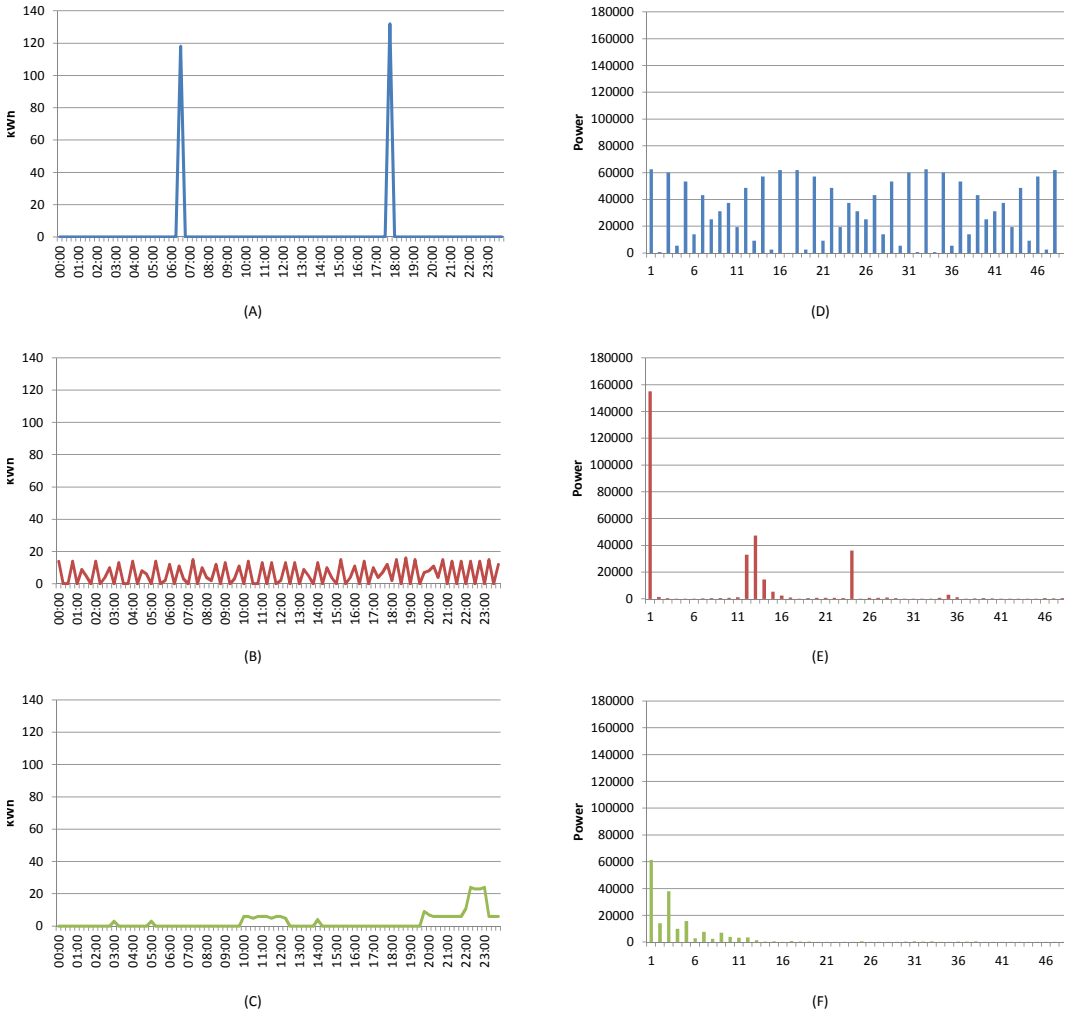


Figure 4.1: Examples series to demonstrate transforming time series from the ElectricalDevices dataset into the power spectrum. (A), (B), and (C) show examples of a kettle, refrigerator and television in the time domain, and (D), (E), and (F) show the transformed versions of these series respectively.

using shapelets, we believe that we can use the best shapelets from a dataset to transform series into a representation that encompasses local similarity in shape. In this section we introduce the core components of the shapelet extraction process; finding a shapelet requires generating a set of candidates, defining a distance measure between a shapelet and each time series, and defining a measure of the discriminatory power of a shapelet. These tasks are described in this section, providing adequate technical background for a shapelet transform that we propose in Chapter 6.

4.2.1 Shapelet Extraction

Given a time series T_i of length m, a candidate shapelet is a subsequence S of length $l \leq m$ that is a contiguous sequence of points from T_i. Therefore any series of length m contains $m - l + 1$ distinct subsequences of length l. The set of all subsequences of length l for T_i can be denoted as $W_{i,l}$. For a set of time series T of length n, the set of all subsequences of length l can be denoted as:

$$W_l = \{W_{1,l}, W_{2,l}, ..., W_{n,l}\}. \tag{4.6}$$

The set of all candidate shapelets for all series in T can then be denoted as:

$$W = \{W_{min}, W_{min+1}, ..., W_{max}\}, \tag{4.7}$$

where $min \geq 3$ and $max \leq m$ represent the minimum and maximum shapelet lengths respectively. It should be observed that W is very large, consisting of $O(nm^2)$.

The generic shapelet extraction algorithm that was originally proposed in [107] is defined in Algorithm 6 (Section 4.2.1).

Algorithm 6 ShapeletSelection (T, min,max)

```
1:  best = 0;
2:  bestShapelet = ∅;
3:  W = generateCandidates(T, min, max);
4:  for l = min to max do
5:     for all subsequence S in W_l do
6:        D_S = findDistances(S, T);
7:        quality = assessCandidate(S, D_S);
8:        if quality > best then
9:           best = quality;
10:          bestShapelet = S;
11: return bestShapelet;
```

4.2.2 Assessing Shapelet Candidates

Two key components in the shapelet extraction algorithm outlined in Algorithm 6 are the *findDistances* and *assessCandidates* functions. The first operation creates an ordered set of distances D_S between a candidate shapelet S and a set of training data T. This is then used in *assessCandidates*, where the quality of S is determined for T by judging how well the shapelet splits the training data. To carry out these tasks, two operations must be defined: firstly, how to measure the similarity between S and the training data T to calculate D_S, and secondly, how to use D_S to calculate the quality of S.

4.2.3 Shapelet Similarity

We denote the Euclidean distance between two subsequences S and R of length l as

$$dist(S,R) = \sum_{i=1}^{l}(s_i - r_i)^2. \tag{4.8}$$

The distance between a subsequence S of length l and time series T_i is the minimum distance between S and all normalised subsequences of T_i of length l, i.e.

$$d_{i,S} = \min_{R \in W_{i,l}} dist(S,R). \tag{4.9}$$

We generate all distances between a candidate shapelet S and all series in T to generate a list of n distances,

$$D_S =< d_{1,S}, d_{2,S}, \ldots, d_{n,s} > . \tag{4.10}$$

Note that since $d_{i,S}$ is a minima, [107] propose speeding up the calculation of $d_{i,S}$ with an early abandon.

4.2.4 Shapelet Quality Measures

The original shapelet algorithm in [107] uses information gain to establish the quality of a shapelet for splitting a dataset. This measure lends itself well to the decision tree classifier that the authors embed shapelets within, as previously seen with C4.5 decision trees in Section 2.4. However, we believe that alternative measures can be used in place of information gain which may be more appropriate in future non-tree based implementations of shapelets. Below we discuss the information gain measure used in [107], and also introduce three alternative quality measures: the F-statistic from analysis of variance, Kruskal-Wallis, and Mood's Median.

Information Gain

Information gain [98] (IG) is a non-symmetrical measure of the difference between two probability distributions. The shapelet finding algorithm in [107] uses IG as the quality measure to assess candidate shapelets. D_S is sorted and the information gain at each possible split point sp is assessed for S, where a valid split point is the average between any two consecutive distances in D_S. For each possible sp, IG is calculated by partitioning all elements of $D_S < sp$ into A_S, and all other elements into B_S. The information gain at sp is calculated as

$$IG(D_S, sp) = H(D_S) - \frac{|A_S|}{|D_S|}H(A_S) + \frac{|B_S|}{|D_S|}H(B_S), \tag{4.11}$$

where $|A_S|$ is the cardinality of the set A_S, and $H(A_S)$ is the entropy of A_S. Entropy is calculated by

$$H(D_S) = -\sum_{c \in classes\{D_S\}} p_c \log_2 p_c. \tag{4.12}$$

The IG $info_S$ of S is calculated as

$$info_S = \max_{sp \in D_S} IG(D_S, sp). \tag{4.13}$$

Analysis of Variance F-statistic

The F-statistic (F-stat) in analysis of variance (ANOVA) is used to test the difference between means from a set of C samples, using the null hypothesis that there is no difference between each of the population means. We believe that F-stat will discriminate between shapelets well since the statistic is based on the ratio of variability between groups to the variability within the groups. High values of the test statistic would indicate that the between-group variability is high, but within-group variability is low. This is ideal for shapelet candidates, as a high F-stat would indicate that the shapelet discriminates between classes well (high between-class variability), but matches well within the class that the shapelet is extracted from (low within-class variability).

Given the set of distances D_S for dataset T and shapelet S, the F-stat is calculated by first separating the distances by class value, such that $D_{S,i}$ contains the distances between S and all members of T with the class value i. Once these groups are established, the F-stat is calculated as:

$$F = \frac{\sum_i (\bar{D}_{S,i} - \bar{D}_S)^2/(k-1)}{\sum_{i=1}^{k} \sum_{d_j \in D_{S,i}} (d_j - \bar{D}_S, i)^2/(|D_S| - k)} \tag{4.14}$$

where C is the number of classes in T, $|D_S|$ is the cardinality of D_S, $\bar{D}_S$ is the overall population mean, and $\bar{D}_{S,i}$ is the mean of the distances to series of class i. Typically ANOVA is used to evaluate whether there is a significant difference between samples by comparing the calculated F-stat to a critical difference value. However in this context, a critical difference value is not necessary, and the F-stat alone can be used as a relative measure of quality between candidate shapelets.

Kruskal-Wallis

Kruskal-Wallis [68] (KW) is a non-parametric test to observe whether data originates from a single distribution. The calculated statistic represents the squared-weighted difference between ranks within a class and the global mean rank. For use with shapelets, KW is calculated for S as

$$KW_S = \frac{12}{|D_S| \cdot (|D_S| + 1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(|D_S| + 1), \tag{4.15}$$

where $|D_S|$ is the cardinality of D_S, k is the number of classes in D_S, R_i is the sum of ranks for class i and n_i is the number of instances of class i in D_S. Note that in order to calculate ranks, D_S must be sorted as it was with IG. However, we believe that KW will be more efficient for shapelet finding than IG because the statistic only needs to be calculated once, rather than for each possible split point in D_S.

Mood's Median

Mood's Median [19] (MM) is a non-parametric test to determine whether the medians of two samples originate from the same distribution. Unlike IG and KW, MM does not require D_S to be sorted, so therefore should be faster. Only the median is required for calculating MM, which can be found in $O(n)$ time using quickselect [54]. The median is used to create a contingency table from D_S, where the counts of each class above and below the median are recorded. The MM statistic is obtained by calculating the Chi-Squared statistic of the table

$$\chi^2 = \sum_{j=1}^{c} \sum_{i=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}, \tag{4.16}$$

where r and c are the rows and columns of the contingency table and o_{ij} and e_{ij} are the observed and expected values of row r, column c respectively.

4.2.5 Example: MPEG7 Data

Though the authors of the original shapelet work in [107] embedded extraction within their classification algorithm, we wish to use shapelets as a way to consider local shape-based similarity between series. To demonstrate how shapelets can achieve this we create an example from the MPEG-7 datasets introduced in Section 3.4. Specifically, we extract the best shapelet from the *Bird/Chicken* problem, which is equivalent to creating a shapelet decision tree with a forced depth of 1. The shapelet is demonstrated in Figure 4.2. The best shapelet is taken from an instance of the *Bird* class, where the subsequence corresponds to the back of the bird. It is intuitive to see how this shapelet discriminates between the class values from observing the example data. The instances that correspond to the *Bird* class each have a smooth back, whereas those in the *Chicken* class all have a large, raised tail. Therefore distance calculations between the shapelet shown in Figure 4.2 and *Bird* instances will yield a small difference, while the distance between the shapelet and cases of the *Chicken* class will result in large distances since there is no location in those outlines that is a good match.

Figure 4.2: An example of the best shapelet extracted from the bird/chicken problem. (A) shows sample data from the problem, and (B) demonstrates the shapelet that was identified as the best candidate for diferentiating the classes within the training data.

4.3 Similarity in Change: Autocorrelation Transform

The final time series transformation that we consider is based on the autocorrelation of time series. We can quantify the autocorrelation of series within a dataset using the autocorrelation function (ACF). The ACF of a series measures the interdependence of attributes, where a positive autocorrelation suggests a form of persistence in the data, while negative autocorrelation suggests high volatility in the data. For example, if a series

has many consecutive readings above the series mean, or many consecutive reading below the mean, this can be seen as persistence within the data as new readings appear to stay in the same state as previous readings. Conversely, if readings above the mean tend to be followed immediately by readings below the mean, this would suggest high volatility within the data and a negative autocorrelation.

An effective way to analyse characteristics of the autocorrelation of a series is to create the correlogram of the series by transforming with the ACF. The correleogram contains the sample autocorrelation coefficients, which measure the correlation between observations at various time lags. As an example, consider the series $T =< t_1, t_2, \ldots, t_m >$. The first coefficient r_1 in the ACF will be at lag 1. Therefore the coefficient is calculated as the correlation between the first $m-1$ readings in T and the last $m-1$ readings in T, and the result will be within the range of $[-1, 1]$. This is calculated as:

$$r_1 = \frac{\sum_{i=1}^{m-1}(t_i - \bar{t}_{(1)})(t_{i+1} - \bar{t}_{(2)})}{\left(\sum_{i=1}^{m-1}(t_i - \bar{t}_{(1)})^2\right)^{1/2}\left(\sum_{i=2}^{m}(t_i - \bar{t}_{(2)})^2\right)^{1/2}} \tag{4.17}$$

where $\bar{t}_{(1)}$ is the mean of the first $m-1$ observations and $\bar{t}_{(2)}$ is the mean of the final $m-1$ readings. However, this calculation can be simplified by approximation when m is reasonably large. The difference between the subsequence means $\bar{t}_{(1)}$ and $\bar{t}_{(2)}$ is relaxed, and the mean of the whole sample is used instead. Furthermore the summations from 1 to $m-1$ and 2 to m can be removed to use a single iteration from 1 to m in the denominator. Therefore we can now approximate r_1 as:

$$r_1 = \frac{\sum_{i=1}^{m-1}(t_i - \bar{t})(t_{i+1} - \bar{t})}{\sum_{i=1}^{m}(t_i - \bar{t})^2}. \tag{4.18}$$

We can then generalise this for any lag value k where $k < m$:

$$r_k = \frac{\sum_{i=1}^{m-k}(t_i - \bar{t})(t_{i+k} - \bar{t})}{\sum_{i=1}^{m}(t_i - \bar{t})^2}. \tag{4.19}$$

One final simplification can be made to the calculation of r_k due to the nature of the data that we use. It is common in TSC to work with series that have been preprocessed

using various normalisation techniques, most notably zero mean and unit variance normalisation. When we implement the ACF transform, we use both of these normalisation approaches on the data before calculating the correlogram of series. As a consequence, we can simplify the calculation of r_k to:

$$r_k = \sum_{i=1}^{m-k} (t_i \cdot t_{i-k}). \tag{4.20}$$

Using this calculation, ACF can be used to transform an input series T into an output sequence of the autocorrelation at l different lag values:

$$R = \{r_2, r_2, ..., r_l\} \tag{4.21}$$

However, ACF is only one method for quantifying the autocorrelation of a series. A further possibility is the partial autocorrelation function (PACF). The PACF of a series describes autocorrelation with linear dependence between attributes removed. For example, if we consider three adjacent attributes in a time series, a, b, and c; if b is highly correlated with a and c is highly correlated with b, c is linearly dependant on a, and any correlation with b may be a residual effect of this relationship. Therefore the PACF measures autocorrelation with this dependence removed. The PACF can be found from the ACF through a series of linear equations.

Given the first p terms of the calculated ACF, there exists a set of parameters $\Lambda_p = (\lambda_1, \lambda_2, \ldots, \lambda_p)$ that satisfy $R_p = \Lambda_p \Phi_p$, where Φ_p is the Toeplitz matrix of the p ACF values:

$$\Phi_p = \begin{bmatrix} 1 & r_1 & r_2 & r_3 & \cdots & r_{p-1} \\ r_1 & 1 & r_1 & r_2 & \cdots & r_{p-2} \\ r_2 & r_1 & 1 & r_1 & \cdots & r_{p-3} \\ r_3 & r_2 & r_1 & 1 & \cdots & r_{p-4} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & r_{p-3} & r_{p-4} & \cdots & 1 \end{bmatrix}. \tag{4.22}$$

By rearranging the previous equation, the set of parameters for the first p terms is be found as $\Lambda_p = \Phi_p^{-1} R_p$. For all values of p, using series length as m and the maximum

lag as l, we have:

$$\Lambda = \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \vdots \\ \Lambda_{m-l} \end{bmatrix} = \begin{bmatrix} \lambda_{1,1} & & & & \\ \lambda_{2,1} & \lambda_{2,2} & & & \\ \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \lambda_{m-l,1} & \lambda_{m-l,2} & \lambda_{m-l,3} & \cdots & \lambda_{m-l,m-l} \end{bmatrix} \tag{4.23}$$

The PACF is given as the diagonal through the matrix, $P =< \lambda_{1,1}, \lambda_{2,2}, \ldots, \lambda_{m-l,m-l} >$. This can be found through solving $m - l$ systems of linear equations. However, Φ is a Toeplitz matrix, since the diagonals within the matrix are constant. This property can be exploited for faster processing by using the Durbin-Levinson algorithm [70, 36], which has a time complexity of $O(n^2)$. In the same manner as we described for the ACF terms, the resulting PACF terms can also be used to represent a time series in the autocorrelation domain.

In both cases with ACF and PACF, we need to set the maximum lag l. By definition, the higher the lag is in the derived terms, the greater the variability. Therefore placing a large restriction on lag will retain only the most explanatory terms. In our implementations of ACF and PACF, we restrict l to either $m/4$ or 100, depending on which value is largest. To demonstrate the application of ACF and PACF to time series data, we include an example in Figure 4.3 using the two class *Caenorhabditis elegans* problem that was introduced in Section 3.5.

The first column in Figure 4.3 depicts the ACF (A) and PACF (C) for the control group in the data. The ACF demonstrates a strong positive trend in the relationship between attributes at different lag values, while observing the PACF shows that with linear dependence removed, there is a strong relationship with a lag of 1. The example taken from the mutant strain has a very different ACF (B), with an initially strong positive correlation that inverts to a negative correlation. Observing the PACF shows that there is a strong positive relationship with lag at 1, but subsequently there is a strong negative relationship with a lag of 2. This would suggest that the movement of the mutant instance is much more erratic than that of the control group worm.

We believe that transforming time series with ACF and PACF will allow classification algorithms to detect change-based discriminatory features between class values. However, this is not an approach that is adopted widely in the literature. The most common approach of using autocorrelation-based approaches with time series data is to fit autoregressive (AR) models to series. This is typically carried out with a different purpose

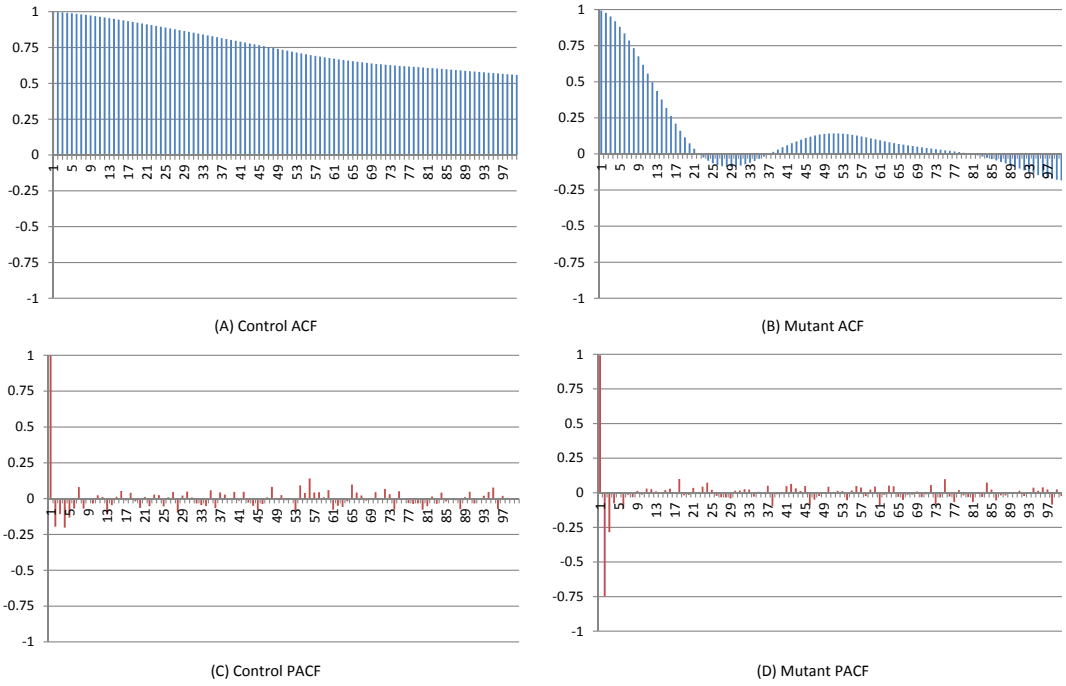


Figure 4.3: Examples of the ACF and PACF transforms on the Worms2 problem.

in mind to TSC however, with the predominant aim of forecasting future values [14]. In cases where AR models have been applied in TSC, the process typically involves extracting parameters after fitting models, and using a distance measure to calculate the similarity between the parameters (for example, with the Euclidean distance [27]).

Chapter 5

Time Domain Classification: Current Benchmarks and a New State-of-the-art

Contributing Publications

- Jason Lines and Anthony Bagnall. Ensembles of elastic distance measures for time series classification. In *Proceedings of the 14th SIAM International Conference on Data Mining (SDM)*, pages 524–532. 2014.
- Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, pages 1–28, 2014.
- Anthony Bagnall and Jason Lines. Technical report: An experimental evaluation of nearest neighbour time series classification. *CoRR*, abs/1406.4757, 2014.

The majority of research in TSC has focused on classification in the time domain. The general consensus among researches is that the current benchmark for TSC is to use nearest neighbour classifiers (NN) with similarity between series evaluated using either the Euclidean distance of Dynamic Time Warping (DTW) with a warping window set through cross-validation. With this in mind, the objectives of this chapter are two-fold. Firstly, we aim to extensively evaluate the utility of using NN classifiers for TSC. A preliminary study is carried out to test the claim that *"simple nearest neighbor classification is exceptionally difficult to beat"* by comparing NN classifiers to various standard

classification algorithms (Section 5.2). We then carry out experiments to address several aspects of applying DTW to TSC problems, such as whether setting warping window or neighbourhood sizes significantly affects error rates (Section 5.3). Once the optimal configuration for DTW is established, an extensive evaluation is carried out to compare the classification performance of NN classifiers using DTW against alternative similarity measures (Section 5.4). We consider a total of 11 distinct NN approaches, with the goal to determine whether DTW is in fact the gold standard for TSC with NN classifiers, or if there are any alternative elastic measures that should be considered as the new benchmark. 2:30

The second aim of the chapter is informed by the first. If NN classification with DTW is still the leading candidate for TSC, or if there is no other dominant approach, we want to test if it is possible to combine the decisions of multiple measures into a single classifier that is significantly more accurate than any of the individual components. We investigate this in Section 5.5 through using simple ensemble schemes to create a new classifier: the *Elastic Ensemble* (EE).

The contributions of this chapter are summarised as follows:

- We demonstrate that NN classifiers with DTW are not outperformed by any alternative classification approach (Section 5.2). This reinforces the view that DTW is hard to beat. However, Euclidean distance with NN classifiers was beaten by a subset of the alternative approaches. Euclidean NN is often used as a benchmark to justify new algorithms; these findings suggest that Euclidean NN should no longer be used as such, and DTW should be the only configuration of NN classifiers that is used for comparison.

- Setting the warping window for DTW 1-NN through cross-validation on training data (DTWCV) is shown to significantly improve the classifier (as first recognized in [90]). We also show that setting the number of neighbours in the NN classifier for TSC may be beneficial (Section 5.3.1), but the support for this is not as strong as setting measure parameters (Section 5.3.2). The results lead to the recommendation that setting measure parameters through CV should take priority over setting neighbourhood sizes, as it may be intractable to search for both parameters on large problems. We therefore advocate the best configuration for approaching TSC problems with DTW is to use a 1-NN classifier, and to set the DTW warping window through cross-validation (DTWCV).

- None of the elastic measures that we assess are significantly more accurate than

DTW (Section 5.4). These results demonstrate that there is no dominant approach for TSC, and the popular approach of DTWCV can still be considered the benchmark approach for TSC.

- Combining the elastic measures using simple ensemble schemes produces a classifier that is significantly more accurate than any of the constituent parts over the full set of 75 datasets and the UCR datasets (Section 5.5.3). Investigation showed that while the overall accuracies of the classifiers were not significantly different, the individual predictions made by each were significantly different (Section 5.5.1). To counter any suggestions of bias, the experiments are repeated using the set of 46 widely-used UCR datasets, and the EE remained significantly more accurate than any competing algorithm. Comparisons to recent work in the literature (Section 5.5.4) lead us to believe that these are the best TSC results ever published, and the EE is the first classifier to significantly outperform DTWCV on the UCR datasets.

5.1 Datasets

To keep results consistent and interpretable, each experiment in this chapter is carried out using the same set of 75 datasets. The names of these are shown in Table 5.1. For more information on specific datasets, please refer to Chapter 3. It is worth reiterating that all datasets are split into training and testing sets, where any parameter optimisation is carried out on the training data only. It is convention to split datasets in this fashion in the TSC literature, and some datasets have been split in a certain manner to avoid introducing bias into the classification problems (for example, the electricity device datasets described in Section 3.2). Additionally, the widely used UCR data repository [65] also organises data in this way, so it is desirable to run experiments using train/test splits to make comparisons possible with other work on the UCR datasets.

5.2 Nearest Neighbour Classification: Hard to beat, or a misconception?

The first aim of this chapter is to test the widely-believed claim that NN classifiers are difficult to beat. We design an experiment to test this by comparing NN classifiers using Euclidean distance and DTW against other standard classification algorithms.

Table 5.1: The 75 datasets used in this chapter, split by problem type. See Chapter 3 for more information on specific datasets.

Image Outline Classification			
DistPhalanxAge	DistPhalanxOutline	DistPhalanxTW	FaceAll
MidPhalanxAge	MidPhalanxOutline	MidPhalanxTW	FaceFour
ProxPhalanxAge	ProxPhalanxOutline	ProxPhalanxTW	WordSynonyms
OSULeaf	Phalanges	yoga	ShapesAll
SwedishLeaf	MedicalImages	Symbols	Adiac
ArrowHead	BeetleFly	BirdChicken	DiatomSize
FacesUCR	fiftywords	fish	Herring
Motion Classification			
CricketX	CricketY	CricketZ	GunPoint
UWaveX	UWaveY	UWaveZ	UWaveAll
Haptics	InlineSkate	ToeSeg1	ToeSeg2
Sensor Reading Classification			
Beef	Car	Chlorine	CinCECG
Coffee	Computers	FordA	FordB
ItalyPower	LargeKitchen	Lightning2	Lightning7
StarLightCurves	Trace	TwoLeadECG	wafer
RefrigerationDevices	MoteStrain	Earthquakes	ECGFiveDays
ElectricDevices	SonyRobot1	SonyRobot2	OliveOil
Plane	Screen	SmallKitchen	ECGThorax1
	ECGThorax2		
Simulated Classification Problems			
ARSim	CBF	SyntheticControl	ShapeletSim
	TwoPatterns	MALLAT	

5.2.1 Experimental Procedure

Two NN classifiers (Euclidean and DTW) and seven standard classifiers are used in this preliminary study. To ensure the simplicity of this prototype experiment, no effort is made to optimise the algorithms; the NN algorithms each use 1-nearest neighbour (1-NN) and parameter-free implementations of the distance measures (i.e. DTW is implemented with a full warping window), while the standard classifiers are implemented with default Weka parameters. The full list of the classifiers used are as follows:

1. 1-NN with Euclidean Distance (NNEuclid)
2. 1-NN with DTW and full warping window (NNDTW)
3. Naïve Bayes (NaïveBayes)
4. Bayesian Network (BayesNet)
5. C4.5 Decision Tree (C45)
6. Random Forest (RandFor)

7. Rotation Forest (RotFor)

8. Support Vector Machine with linear kernel (SVML)

9. Support Vector Machine with quadratic kernel (SVMQ)

The null hypothesis is that there is no difference between the average accuracies of the NN classifiers and the standard classifiers. If this is not disproved, it will demonstrate the effectiveness of simple NN classifiers for TSC. However, if any of the standard classifiers are found to be significantly more accurate than the NN classifiers, the null hypothesis can be rejected and the claim that NN classification is hard to beat must be false.

5.2.2 Results

The results of the 9 classifiers over the 75 datasets are summarised in the critical difference diagram in Figure 5.1. The full results can be found on the supporting website for this thesis [73].

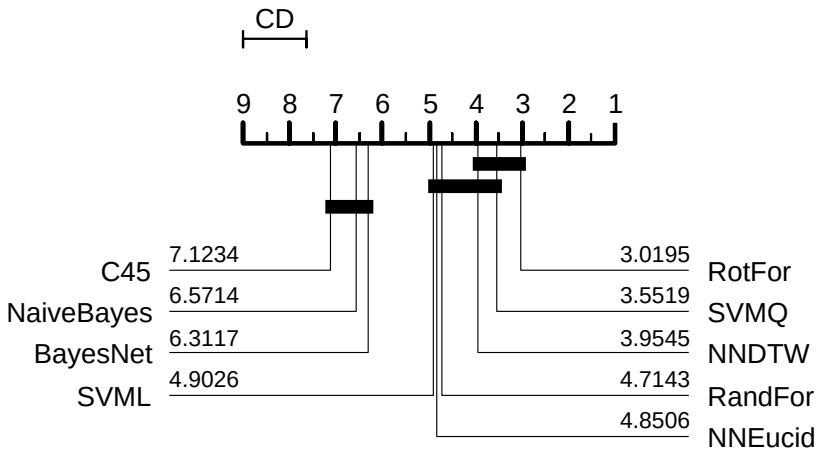


Figure 5.1: A critical difference diagram to compare the test classification performance of NN classifiers with other standard classification algorithms.

.

The CD diagram in Figure 5.1 shows the average rank of each classifier over the 75 datasets. The classifiers are organised into *cliques*, depicted by solid black lines, where there is no significant difference between classifiers. The diagram shows that the top three ranked classifiers are RotFor, SVMQ and NNDTW respectively. Crucially, these

three classifiers are within the same clique, demonstrating that no classifier significantly outperformed NNDTW. This result is critical as it means the null hypothesis cannot be rejected and confirms that NN classification is *hard to beat*. It is important to note however that NNEuclid was not within the top clique, and the average rank was significantly different to the rotation forest. This has important implications for TSC research as NN classification with Euclidean distance and Dynamic Time Warping are often discussed in the same standing; these results show that NNEuclid was outperformed by a classifier using default Weka parameters. Therefore this suggests that using the Euclidean distance with NN should no longer be considered a benchmark in TSC, and work should be compared against DTW.

5.3 Configuring Distance Measures with Nearest Neighbour Classifiers

The preliminary study justified the use of NN classifiers with DTW for TSC. However, this investigation did not address the best solution for implementing NN classifiers with similarity measures. Two obvious questions arise:

1. Is it beneficial to set the number of neighbours for NN classifiers on TSC problems?
2. Should distance measures be parameterised when used with NN classifiers for TSC? i.e. should DTW be implemented with an unconstrained search, or does setting the warping window affect accuracy significantly?

This section addresses these two questions and makes recommendations on the optimal solution for TSC with NN classifiers. This is investigated through using Euclidean Distance and DTW with all of the 75 datasets listed in Section 5.1. Various DTW-based measures are also included for comparison, including derivative and weighted variants of DTW. Seven measures are used in total: Euclidean Distance (ED), DTW with a full window (DTW_{R1}), DTW with window set through CV (DTW_{Rn}), Derivative DTW with full window (DDTW_{R1}), Derivative DTW with window set through CV (DDTW_{Rn}), Weighted DTW with weights set through CV (WDTW), and Weighted Derivative DTW (WDDTW) with weights set through CV.

A common method for improving the accuracy of NN classifiers is to set the number of neighbours k through cross-validation on the training data. This process adds an overhead in the complexity of the algorithm as the distance matrix of the training data must be calculated in order to find k. This overhead is increased dramatically when also

cross-validating for other parameters (such as window sizes for DTW) as each parameter option will likely create different distance matrices. In this investigation, each classifier is given 100 model selections for each parameter; k is set from 1 to 100, and the warping window size/weights start at 1% and increase to 100% in increments of 1%. The number of training experiments required to set parameters for each of the measures is summarised in Table 5.2.

Table 5.2: The distance measures used in the investigation of setting k for NN classifiers with associated training costs. For example, DTW_{Rn} requires that k (100 options) and the window size r (100 options) are both set. Using different window widths will create different distance matrices, so must be recalculated for each combination of k and r ($100 \times 100 = 10,000$ training experiments for each dataset).

Distance Measure	k Options	Parameter Options	Training Experiments per Dataset
ED	100	1	100
DTW_{R1}	100	1	100
DTW_{Rn}	100	100	10,000
DDTW_{R1}	100	1	10,000
DDTW_{Rn}	100	100	10,000
WDTW	100	100	10,000
WDDTW	100	100	10,000

Clearly there is a large overhead for setting both k and measure parameters through CV; setting both would require 100×100 distinct training experiments for a single dataset, resulting in 750,000 training experiments for all 75 datasets. Obviously in cases such as this, the overhead could only be justified if the classifier is significantly improved by setting both parameters. The goal of this section is to clarify this by investigating whether setting k and measure parameters through cross-validation significantly improves accuracy of NN classifiers.

5.3.1 Setting the Number of Neighbours

The first series of experiments is designed to investigate the effect of setting the number of neighbours k in NN classifiers for TSC. 7 variants of the Euclidean distance and DTW were used with NN classifiers (as detailed in Section 5.3). Each classifier had 100 model selections for measure parameters (where applicable) and 100 options for k. For a given dataset, each measure used leave-one-out cross-validation (LOOCV) experiments on the training data to identify the best parameter and k combination, where the best was selected as the combination that resulted in the highest training accuracy. The results

of a direct comparison between 1-NN and k-NN implementations of the 7 classifiers are listed in Table 5.3. Win/tie/loss counts are recorded, along with P values from a two-tailed t-test and Wilcoxon rank-sign test to test for significant differences between using 1-NN and k-NN. These results were originally recorded for the technical report in [6].

Table 5.3: Win/tie/loss counts for using classifiers with 1-NN vs. k-NN. The table also includes p values calculated from performing t-tests and Wilcoxon Signed-Rank tests on the results.

Classifier	Mean kNN Improvement	1-NN Better	Tie	kNN Better	P Value (T-Test)	P Value (Rank-Sign)
ED	0.165%	11	48	18	0.359	0.175
DTW_{R1}	0.281%	11	38	28	0.245	0.036
DTW_{Rn}	1.265%	18	37	22	0.000	0.380
$DDTW_{R1}$	0.733%	8	36	33	0.036	0.004
$DDTW_{Rn}$	0.174%	12	42	23	0.320	0.104
WDTW	6.668%	16	36	25	0.000	0.199
WDDTW	0.799%	12	38	27	0.006	0.010

The t-test in Table 5.3 shows that no significant difference is detected for ED, DTW_{R1} and $DDTW_{Rn}$ at the $p \leq 0.05$ level when using 1-NN against k-NN. Of the other results, WDTW has by far the highest average improvement in accuracy when considering k neighbours. On closer inspection, it was found that there were abnormally large differences on the MiddlePhalanxOutlineAgeGroup and MiddlePhalanxTW datasets of approximately 15% and 11% respectively. These datasets are derived from the same data, so it is likely that there is an underlying relationship that WDTW could not detect with only one neighbour. To identify whether these results skewed the analysis, Wilcoxon signed-rank tests were also carried out as a the non-parametric alternative. The result of this showed that the improvement in WDTW was no longer significant with this test, and DDTW was the algorithm that showed the most significant improvement. However, there is no unanimous result or compelling evidence to suggest that both k and measure parameters should be set, especially in light of the orders of magnitude extra training that is involved in cross-setting two parameters. This potentially leaves a choice; if we only have the resources to set measure parameters *or* set k, which should we favour? The next experiments investigates the claim in [90] that states setting window sizes for DTW significantly improves classification accuracy.

5.3.2 Parameterising Distance Measures

The results from Section 5.3.1 can be reused to formulate a new test. The previous experiment investigated the difference between 1-NN and k-NN implementations of the distance measures. In this experiment, we wish to compare parameterised and non-parameterised measures. Since WDTW must have a weight set in the processing of the algorithm, we focus specifically on full DTW vs. DTW with warping set through cross-validation, and compare results between the two using 1-NN implementations. Reusing the results of DTW_{R1} and DTW_{Rn}, the graph in Figure 5.2 summarises the difference in test accuracy between the two implementations across the datasets.

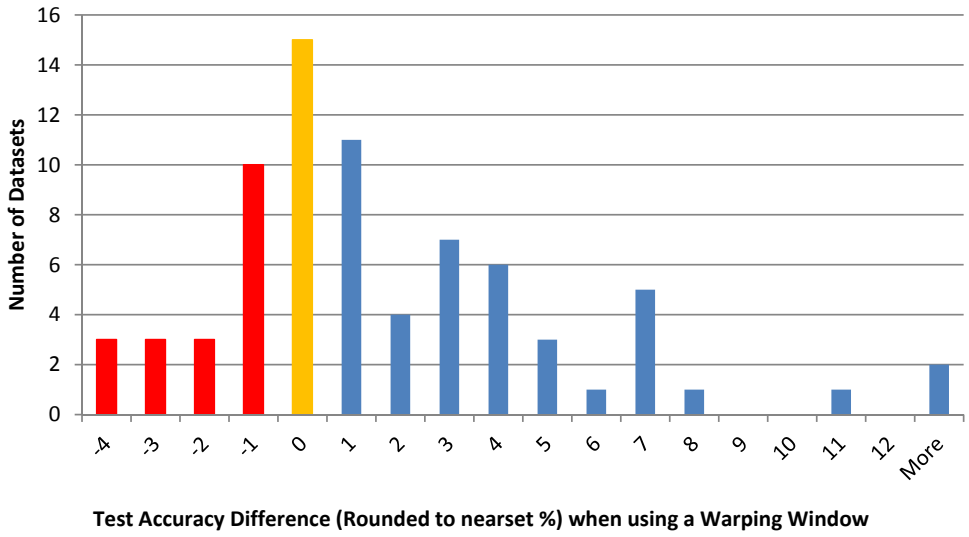


Figure 5.2: A histogram to summarise the differences between using full-windowed DTW against setting the warping window through cross-validation. Positive results (blue) indicate datasets where better performance was obtained using windowed DTW, and negative results (red) indicate where the full-window performed better.

The results demonstrate a significant advantage in test performance when using a window set through cross-validation compared to using the full window. The full window wins on 19 datasets, but the parametrised version wins on 41 datasets. This result is significant according to a t-test and Wilcoxon-signed rank test. Comparing WDTW to DTW with a full warping window exhibited very similar results, as did repeating the comparison using the derivative equivalents of DTW and WDTW. The support for setting parameters is much stronger than setting k, as all classifiers were significantly

improved through setting measure parameters.

5.3.3 Concluding Remarks

The objectives of this section were to answer two questions:

1. Is it beneficial to set the number of neighbours for NN classifiers on TSC problems?
2. Should distance measures be parameterised when used with NN classifiers for TSC?

The first question was answered in Section 5.3.1. The evidence suggested that setting k tended to improve classifiers, but the improvement was not always significant.This casts doubt over the merits of cross-validating for k due to the very large complexity overheads introduced for measures that require parameters. What was evident however is that over the 75 datasets tested, no 1-NN classifier significantly outperformed the equivalent k-NN classifier with the same distance measure. Therefore cross-validating for k appears likely to improve classifiers, and there is no evidence to suggest doing so would decrease classification accuracy. Therefore the recommendation would be to set k if it is possible, but doing so would not necessarily lead to better results.

The second question was answered in Section 5.3.2. The support for the answer of this question was unambiguous; each of the DTW-based measures that were tested were significantly more accurate when parameters were set though cross-validation, as opposed to using the non-parametrised equivalents. This was detected by both t-tests and Wilcoxon-signed rank tests, providing a stronger argument than that of the investigation into setting k. There is evidence to support that setting both of these values improves classifiers, but the training overhead created by searching for both is very large and may become intractable for large problems. We could potentially speed-up searching for k using approaches such as gradient descent or other heuristic searching algorithms, but it should be noted that this is equally possible for finding measure parameters. Therefore our overall recommendation for training NN classifiers on TSC problems would be to prioritise setting similarity measure parameters over setting k, since the results are definitive for setting the former, while the support is not as strong for the latter. Therefore our recommendation is that setting similarity measure parameters should be prioritised over setting k when building classifiers for TSC problems. This conclusion influences the remaining experiments in this chapter; all subsequent NN classifiers are implemented with $k = 1$ and similarity measure parameters are set through CV on the training data (where applicable).

5.4 Comparison of Elastic Distance Measures

The previous work in this chapter has focused on demonstrating the effectiveness of NN classifiers for TSC problems. Section 5.2 demonstrated the potency of the classifiers built with DTW when compared to other leading classification techniques, while Section 5.3 discussed the best configuration for initialising NN classifiers for TSC problems. With this evidence to support NN approaches and guidance of how to configure them with DTW, an obvious question arises: is DTW the best similarity measure for NN classifiers, or are the alternatives that are better for TSC?

Several alternative distance measures for comparing time series have recently been proposed and evaluated on TSC problems. These include variants of DTW, such as weighted and derivative DTW (as previously seen in this chapter), and edit distance-based measures, including Longest Common Subsequence distance (LCSS), Edit Distance with Real Penalty (ERP), Time Warp Edit Distance (TWE), and Move-Split-Merge (MSM) (see Section 2.3). These measures can be considered to be *elastic*, as they can compensate for potential localised misalignment in the time domain.

The objective of this section is to extend the work in [35] by evaluating various elastic distance measures for TSC to determine whether any alternative measures outperform DTW. This is achieved through using the insight learned from Section 5.2 and Section 5.3 to build 1-NN classifiers with various elastic similarity measures. The names of the classifiers and the abbreviations used throughout this section are listed in Table 5.4.

5.4.1 Elastic Measure Experimental Design

All datasets are split into training and test sets and classifiers are only exposed to the training data when being built. All measures are implemented with 1-NN classifiers, and all parameter optimisation is carried out on the training data only (i.e. no validation on the test data). As with the experiments in Section 5.3, each measure is given 100 model selections in cases where parameters must be set; ED, DTW, and DDTW do not require parameter setting; DTWCV, DDTWCV, WDTW, and WDDTW require cross-validation to find window sizes between 1% and 100% in steps of 1%; LCSS requires two parameters, which are derived from the parameters used in [35] to create 100 distinct options; ERP requires two parameters, which are also derived from the parameters used in [35] to create 100 distinct options; TWE requires two parameters, which are resampled into 100 options from the suggested values in [79]; and MSM requires a single parameter, where the range of values is created by resampling 10 suggested parameters from the source code in [101] to obtain 100 values. The full list of parameter options are omitted

Table 5.4: A list of the classifiers used for the elastic measure comparison experiments with associated abbreviations. Each is implemented with a 1-NN classifier, and all parameter setting is performed through LOOCV experiments on the training data only (see Section 2.3 for specific details on each measure).

Classifier	Abbreviation	Parameters Required
Euclidean Distance	ED	-
DTW, full window	DTW	-
DTW, CV Window	DTWCV	warping window
Weighted DTW	WDTW	weighting factor
Derivative DTW, full window	DDTW	-
Derivative DTW, CV window	DDTWCV	warping window
Weighted Derivative DTW	WDDTW	weighting factor
Longest Common Subsequence	LCSS	band size, similarity threshold
Edit Distance with Real Penalty	ERP	band size, similarity threshold
Time Warp Edit Distance	TWE	stiffness, operation cost
Move-Split-Merge Distance	MSM	operation cost

for brevity, but can be found on the website accompanying this thesis [73].

5.4.2 Classification Results

The classification results of the elastic 1-NN classifiers on the test data are summarised by the critical difference diagram in Figure 5.3 (the full results available on the website accompanying this thesis [73]). The diagram shows that there is no classifier that significantly outperforms all others. There are four cliques, where the top clique contains all but ED, DDTW and DTW. This is an interesting finding, since these are the only measures that did not have any parameter optimisation. This reinforces our earlier findings when investigating whether to set measure parameters in Section 5.3 where we demonstrated that setting parameters significantly improved the measures. The overall result of this experiment demonstrates that there is no significant difference between any of the parametrised elastic measures over these datasets, and importantly, none of the alternatives outperform DTWCV. Two main conclusions can be drawn from this result; firstly, the conclusion of Section 5.2 has been reinforced as Euclidean distance was beaten by all of the parameterised elastic measures, while DTWCV was not outperformed by any. Therefore DTW with a warping window set through cross-validation is a good benchmark, but the Euclidean distance does not offer competitive performance. Secondly, setting warping windows through cross-validation improves DTW-based algorithms, as

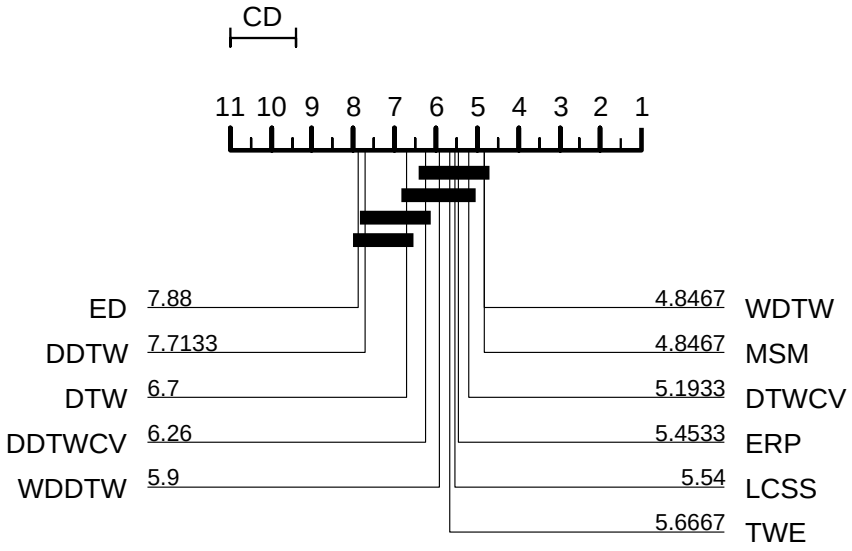


Figure 5.3: The average ranks for eleven 1-NN classifiers on the 75 data sets. The critical difference for $\alpha = 5\%$ is 1.6129.

first observed by [90] and concluded in Section 5.3. It should be noted that while the datasets used in these experiments were not collected with any domain or agenda in mind, there is bias in the type of problems that are considered due to the abundance of certain problem types. Much of the work in the TSC literature is evaluated using the UCR datasets [65] however, allowing the opportunity to compare approaches that are evaluated on this large set of common problems. Therefore to ensure that our conclusion remains consistent with the literature as a whole, the critical difference diagram from Figure 5.3 is recalculated in Figure 5.4 using only the UCR datasets. The conclusions remain consistent and valid.

These results do not lend any weight to one elastic measure over another, suggesting that the standard practice of using DTWCV is still a worthy benchmark. It is by far the most widely adopted measure, and no alternative has been proven to be significantly more accurate over these experiments. This leads to an obvious question: if there is nothing to choose between the measures in terms of classification accuracy, is there any other reason to use one measure over another, or should researchers continue to use DTW?

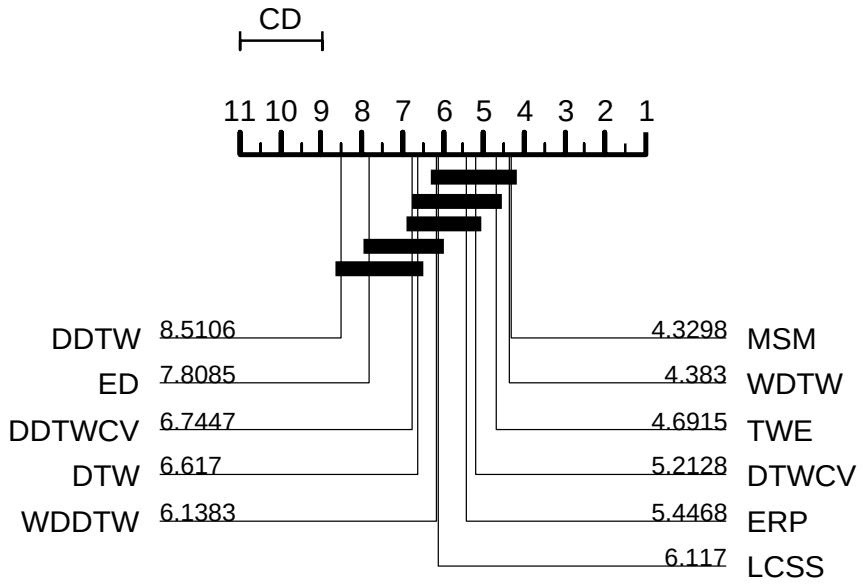


Figure 5.4: The average ranks for eleven 1-NN classifiers on the 46 UCR datasets only. The results demonstrate a very similar pattern to those in the critical difference diagram of Figure 5.3 for the full 75 datasets. Critically, these results still demonstrate that no elastic measure significantly outperforms DTWCV.

5.4.3 A Priori Detection of the Best Measure

The evidence suggests there is no dominant elastic measure that outperforms DTWCV in terms of overall classification accuracy. However, if it were possible to detect which measure would be best for a given dataset in advance, this issue would become irrelevant. The optimal classifier could always be identified, removing the need for finding a single approach to use. Using a critical difference diagram to demonstrate the effect of always selecting the best classifier would not be useful, since the classifier should always simply have a rank of 1. To demonstrate this effect, the average classification accuracies for the eleven elastic classifiers on the 75 datasets are reported in Table 5.5, with an added column to show the difference of each average accuracy to the theoretical classifier that would be achieved from finding the best classifier on each dataset.

The TOP classifier is for information only as the procedure for selecting results obviously introduces bias because accuracy is simply selected from the best classifier on the test data. To create a fair classifier, the best classifier would need to be determined *a priori*. Therefore the best classifier for test classification must be predicted from the training data only. Since the measures have parameters set through cross-validation on the train-

Table 5.5: The average test accuracies of each elastic classifier over the 75 datasets, with an added entry for the theoretical best classifier (TOP) that would be achieved by always selecting the best classifier on each of the 75 problems.

Classifier	Average Accuracy	Difference to TOP
ED	75.75%	9.67%
DTW	79.13%	6.29%
DTWCV	81.44%	3.98%
WDTW	82.09%	3.33%
DDTW	71.97%	13.45%
DDTWCV	76.73%	8.69%
WDDTW	77.19%	8.23%
LCSS	79.50%	5.93%
MSM	82.43%	2.99%
TWE	82.15%	3.27%
ERP	81.08%	4.35%
TOP	85.42%	-

ing data, training accuracies are available for all classifiers. This data can be used to select a classifier on the basis of it performing better than all alternatives on a given dataset, so if training accuracy is a perfect indicator of test classification performance, the best classifier will always be found. This is investigated using Texas Sharpshooter plots, as introduced by [8]. These graphs show the ratio of training accuracy versus test accuracy for a classifier on all datasets by using DTWCV as a benchmark. The desired outcome is that there should be a strong correlation within the ratios, consisting of many true positives and true negatives; in other words, if the training accuracy of a classifier indicates that it is better than DTWCV, the test accuracy should be proportionally superior (and vice versa). Texas Sharpshooter plots for the four highest ranked classifiers in Section 5.4 are compared to DTWCV in Figure 5.5.

The number of datasets with results consisting of true positives or true negatives for LCSS, WDTW, TWE, and MSM are 38, 31, 35, and 49 respectively. This means using training accuracy to predict whether the classifier will outperform DTWCV on the test data is no better than randomly guessing for three of the four classifiers on these 75 datasets! If there is a difference between the classifiers, there is too much bias and/or variance in the training accuracies to detect it. The conclusion is that while training accuracy is a good indication of test classification, it is not sufficient for discriminating between classifiers on these problems.

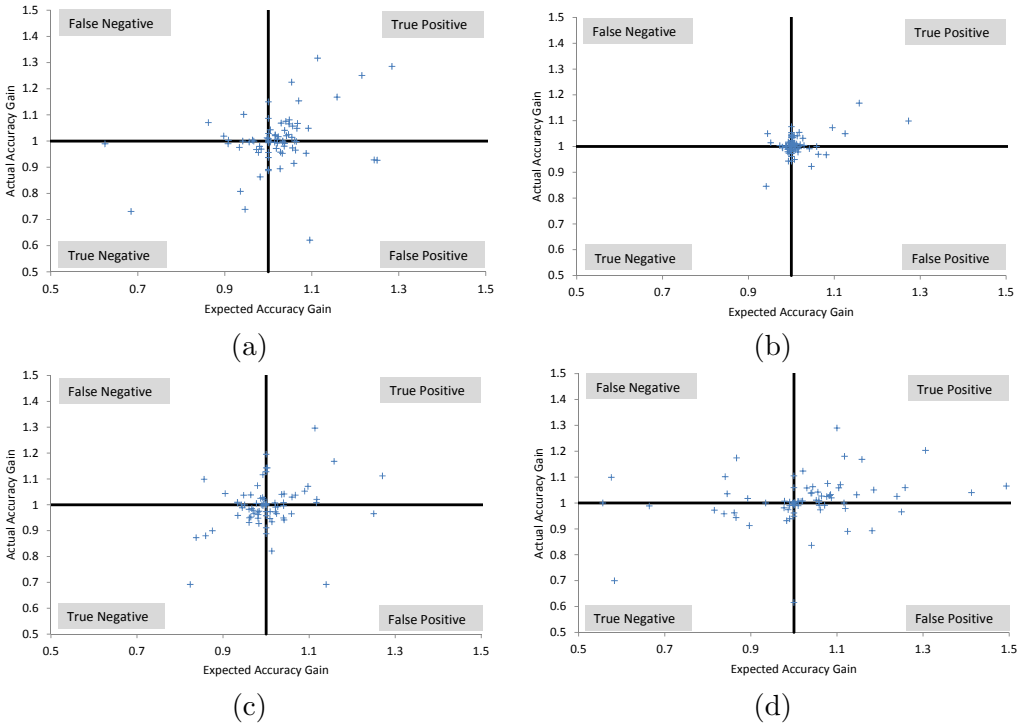


Figure 5.5: Texas sharpshooter plots of DTWCV vs. (a) LCSS (b) WDTW (c) TWE and (d) MSM. The points on the plots represents results for each of the 75 datasets, where expected accuracy gain is calculated as the ratio of the each measure vs. DTWCV on the training data, and the actual accuracy gain is calculated as the ratio between the measures and DTWCV on the test data. If training CV accuracies of the measures offer a good *a priori* indication of test performance, we would expect there to be few false results. However, there is no clear trend of true positives or true negatives with any of the four classifiers, so simply relying on the training CV accuracies to determine which measure will be most effective in advance is not a reliable strategy.

5.4.4 Timing Comparison

Classification accuracy is not the only criteria that is used to compare algorithms. The time complexity of new algorithms can be critical to performance under certain conditions, therefore it is reasonable to also consider the time complexity of each elastic measure when considering alternatives to DTW. Since the NN approach is consistent to the implementation of each measure, this will be omitted from this discussion.

Of the measures used in the experiments of Section 5.4, ED has the lowest time complexity. This is no surprise since the similarity of two series with ED is measured by simply performing a point-wise comparison, thus has $O(n)$ time complexity for series of length n. However, the results in Section 5.4 demonstrated that ED was significantly less accurate than the parameterised elastic measures, so ED is not a viable candidate to replace DTW. The remaining algorithms each have a basic time complexity of $O(n^2)$, but this calculation is slightly confounded when parameter options are introduced. For example, using a warping window with DTW reduces the complexity to $O(nr)$, where r is the amount of warping allowed.

In general, since they share the same time complexity there is little to choose from when comparing the runtime of the different measures. A simple timing experiment was carried out to demonstrate this. For each dataset, the measures were used to calculate the distance between two series. This was repeated 10 times for each of the 100 model selections available to each measure, and the overall median time was taken for each measure/dataset combination. The result of this is shown in Figure 5.6. It is clear that ED is much faster than the competitors, and it is interesting to note that TWE does not scale as well as the other elastic measures. This may be caused by the parameters that are considered, which could cause the algorithm to have a wider average search space than the other measures. However, since the underlying complexities of the measures are still $O(n^2)$, there is still no conclusive evidence to support one elastic measure over the alternatives.

5.5 Combining Elastic Measures: The Elastic Ensemble

Three main conclusions have been drawn in this chapter. Firstly, the belief that DTW with NN classifiers is hard to beat has been supported through experimental comparison to other classification approaches (Section 5.2). Secondly, the question of how best to parametrise NN classifiers for TSC was answered by demonstrating that setting measure parameters through CV leads to significantly more accurate classifiers (Section 5.3).

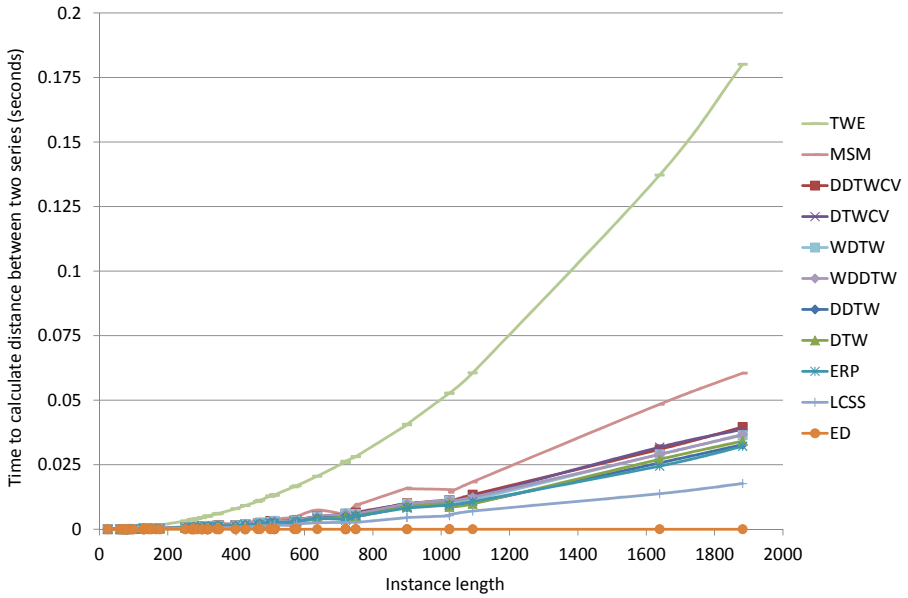


Figure 5.6: A graph to demonstrate the difference in calculation times between each of the elastic measures.

Thirdly, it was shown that while there are numerous alternative elastic similarity measures for time series, none that were tested significantly outperformed DTW with a warping window, nor do any of the viable alternatives provide other advantages such as being able to detect the best method *a priori* or having better runtimes (Section 5.4).

With the goal to find the best approach towards time domain classification for TSC, these conclusions present one final question. As there is no evidence to suggest using an alternative measure in place of DTW, should the alternative measures be disregarded, or is there a way to combine them to create an algorithm that improves upon DTW? We investigate this by firstly evaluating the measures to determine whether they produce different predictions. While the classification results indicated that there was no significant difference between the top elastic measures, it is still possible that they produce significantly different predictions. If this is the case, a number of simple ensemble strategies can be investigated to combine predictions of the individual measures.

Through proportional weighting of elastic measures according to training accuracies, a novel ensemble classifier is created: the Elastic Ensemble (EE). The EE is shown to be significantly more accurate than any of the individual elastic measures, including DTWCV, over the full set of 75 datasets. Furthermore, this result is repeated on the UCR problems, producing what is believed to be the first set of results that are significantly more accurate than DTW on the UCR data.

5.5.1 Measure Divergence

A key component of any ensemble scheme is diversity; if the constituents do not produce significantly different predictions, the overall output of an ensemble will likely be no better than the individual parts. However, if the classifiers are approximately equal in accuracy but divergent in predictions, it may be possible to reduce variance and produce a classifier that is more accurate overall.

A pairwise McNemar's test was used between each classifier to test the null hypothesis that the underlying models are the same across the 75 datasets. Using the 5% level, Table 5.6 shows the number of datasets where the instance predictions were detected to be significantly different between classifiers.

Table 5.6: The number of datasets where the individual classifiers were measured to be significantly different from one other for the top 6 ranked classifiers.

	DTWCV	WDTW	LCSS	TWE	ERP	MSM
DTWCV	0	1	30	22	14	23
WDTW	1	0	26	21	12	22
LCSS	30	26	0	24	23	18
TWE	22	21	24	0	19	17
ERP	14	12	23	19	0	20
MSM	23	22	18	17	20	0

There is only a single dataset where WDTW is significantly different to DTWCV, which perhaps is to be expected as they are different solutions to constraining the same underlying algorithm. However, the DTW-based approaches are both significantly different to LCSS, TWE, and MSM on over 20 datasets each. A similar pattern is repeated between the other classifiers too. Clearly, while the overall accuracies of the classifiers were not significantly different, there is clear diversity in predictions on many datasets. This outcome motivates the potential for combining predictions to create a superior classifier.

5.5.2 Ensemble Design

An ensemble of classifiers is a set of base classifiers, where individual decisions are combined to classify new examples (see Section 2.5 for more discussion). The ensemble classifier that we propose is intentionally intuitive and simple to implement, and does not employ any data pre-processing. The ensemble consists of the 12 elastic classifiers from Section 5.4, hence we name it the Elastic Ensemble (EE). For a given dataset, the only information used to inform a prediction by the EE is the individual training accuracies of the constituent classifiers, which are recorded as a by-product of the parameter

optimisation process outlined in Section 5.3. Constituents that do not require optimisation (e.g. ED, DTW, DDTW) require a single LOOCV experiment on the training data to produce a representative accuracy. Furthermore, all ties are split randomly. Using this outline, the EE was defined and tested with four simple ensemble strategies: *Equal*, *Best*, *Proportional*, and *Significant*.

Equal

Equal is the simplest ensemble strategy available for the EE; it disregards all training information, and simply gives each classifier an equal vote. The votes are summed to find the majority class value, and ties are split randomly.

Best

Best is the polar opposite of *Equal*, placing complete faith in the training results by picking only the best classifier across training for test classification. This is effectively the same strategy as discussed in Section 5.4.3, which stated that if training accuracies are a good identifier of test accuracy with low bias and variance, the best classifier for a dataset should be detectable in advance.

Proportional

Proportional is highly informed by the training accuracies. All classifiers are included in the ensemble, but votes are weighted proportionally according to training performance. For example, if a classifier reported 69% accuracy in training, it would be given 0.69 of a vote in the ensemble. This scheme is applied to all constituents, and votes are combined to pick the decision with the highest weighting.

Significant

Significant is a combination of the *Best* and *Proportional* schemes. The best classifier on the training data is identified, and McNemar's test is performed between the best and all other constituents. Under the null hypothesis that there is no significant difference between the classifiers, any classifier where the null is rejected is given a vote of 0 (effectively removing it from the ensemble). Any classifier that is not removed, has a weight set according to the Proportional scheme. The aim of this scheme is to remove classifiers in situations where they are clearly inferior. For example, ED would not be expected to

perform well on an image outline classification problem where data is not rotationally aligned, so including the measure within the EE could potentially skew the voting.

5.5.3 Elastic Ensemble Results

The results of the EE on the 75 datasets are summarised in the critical difference diagram in Figure 5.7. The full table of results can be found on the website accompanying this thesis [73].

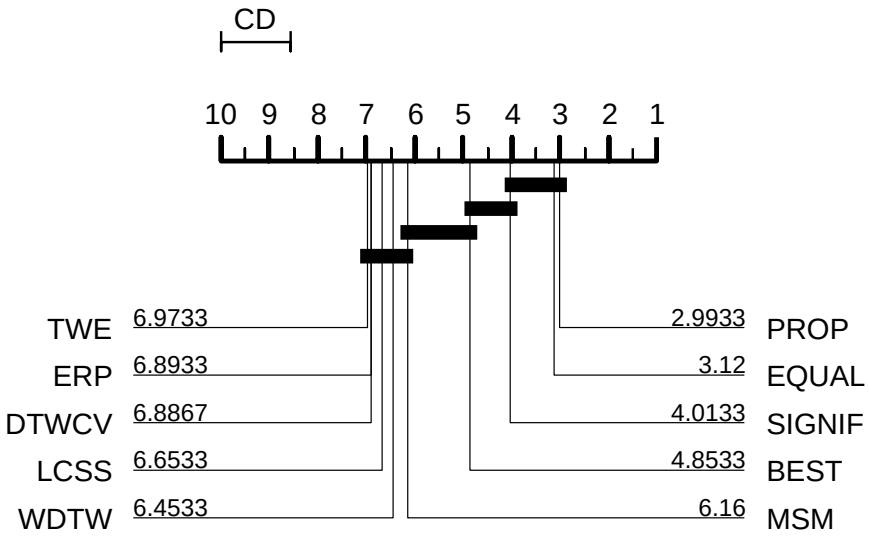


Figure 5.7: The average ranks for the six highest ranked individual classifiers and four ensemble techniques across the 75 datasets. It should be noted that the lower-ranked classifiers (ED, DTW, DDTW, DDTWCV, WDDT) have been removed for clarity, but are still included in the ranking calculations.

The three ensemble techniques *Equal*, *Significant* and *Proportional*, are significantly better than all of the individual classifiers. *Best* does not significantly outperform MSM however. This reinforces the conclusions of Section 5.4.3 and reiterates the danger of placing too much emphasis on training cross-validation measures. While better than any single measure, the attempted use of an inclusion threshold in *Significant* did not provide any improvement on the *Proportional* weighting. To further underpin the performance of the EE a scatter plot is shown in Figure 5.8 to demonstrate the results of the best ensemble scheme (the *Proportional* ensemble) against DTWCV, where each point reflects the results on a single dataset.

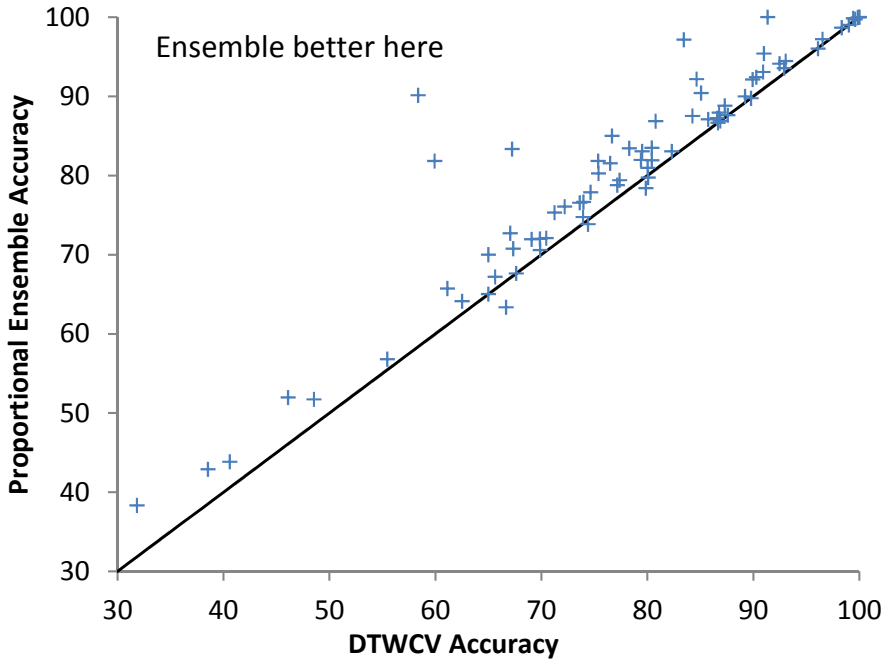


Figure 5.8: A scatter plot to show the representative performance of the *Proportional* EE vs. DTWCV over the full set of 75 test datasets. Plots created for the other measures, such as LCSS, TWE, and MSM, show a very similar result.

In addition to superior performance on the 75 datasets, EE is significantly better than all individual approaches on the UCR datasets too. We believe that this is the first time a classifier has significantly outperformed DTWCV on the UCR data. To counter any accusations of implementation bias, a final critical difference diagram is presented in Figure 5.9. This diagram depicts the results of the EE (*Proportional*) on the original 19 UCR datasets that were used when proposing MSM and TWE in [101] and [79] respectively (both originally used 20 datasets but the ECG_200 dataset was found to be flawed in [7], making it trivial to get perfect accuracy. Therefore this dataset has been removed from the analysis as discussed in Section 3). The results reported for TWE and MSM are taken directly from the work that introduced them, while the results for ERP and LCSS have been taken from the experiments in Section 5.4. Firstly, the author's own results show no critical difference between DTWCV and WDTW, TWE or MSM. This is despite the fact that the results reported for WDTW include allowing the algorithm half the test data for model selection (an advantage they did not give to DTWCV). Secondly, the ensemble results are significantly better.

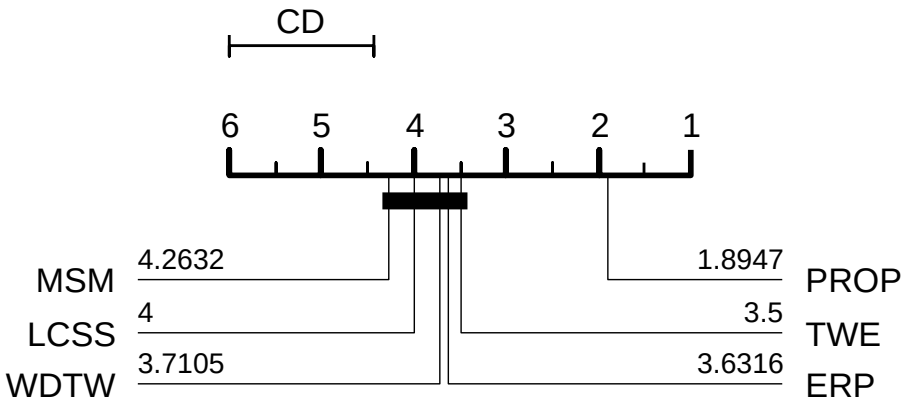


Figure 5.9: Ranks of five classifiers and the EE on the original 19 UCR data sets (omitting ECG).

5.5.4 Elastic Ensemble vs. Other Approaches

There are currently 47 UCR data sets available, and the majority of recent TSC research has included evaluation on at least a subset of these datasets. While it is not possible to compare all algorithms over all datasets, it is possible to compare the EE to other approaches on the common datasets. Of the 47 UCR datasets, 2 are not in the general release (car and plane) and ECG200 is removed from our experiments as previously discussed. Therefore, the greatest number of results that we can compare against is 44 when assuming that results are published for all datasets. For the following analysis, we test the difference between pairs of algorithms using a Wilcoxon signed rank test with α set to 1%. Additionally, a full list of the results is presented in Table 5.7 and Table 5.8, and is also available from the supporting website [73].

- Time Series Forests (TSF) [34] were evaluated on all 44 UCR data sets. The ensemble is better on 29, worse on 15. The difference is significant.
- The time series based on a bag-of-features representation (TSBF) described in [11] is evaluated on 44 UCR data sets. The ensemble outperforms TSBF on 27 data sets and is worse on 17. The difference is significant.
- An adjustment to Euclidean distance that compensates for the complexity of the series [9] (CID) was evaluated on 42 UCR data sets. The elastic ensemble was more accurate on 35 of these. The difference is significant.
- The Bag of Patterns (BoP) approach [72] is evaluated on 19 UCR data. The ensemble has significantly lower error than BoP, winning on 15, tieing on 2, and

Table 5.7: The full results of the state of the art in TSC on the UCR datasets (part one of two). The results are split into two parts for readability; part one includes results for Euclidean Distance (ED), DTW with window set through CV (DTWCV), Time Series Forests (TSF) [34], Time Series Bag of Features (TSBF) [11], Complexity-Invariant Distance (CID) [9], and the Proportional Elastic Ensemble (PROP). Dashes indicate where results were not available. The results are reported as error rates and continued in Table 5.8, and the best result on each dataset across both tables is shown in bold.

Dataset	ED	DTWCV	TSF	TSBF	CID	PROP
Adiac	0.389	0.391	0.261	**0.245**	0.379	0.353
Beef	0.467	0.467	0.3	0.287	0.467	0.367
Car	0.267	0.233	-	-	-	**0.167**
CBF	0.148	0.004	0.039	0.009	**0.001**	0.002
ChlorineConcentration	0.35	0.35	0.26	0.336	0.351	0.36
CinC_ECG_torso	0.103	0.07	0.069	0.262	0.054	0.062
Coffee	0.25	0.179	0.071	0.004	0.179	**0**
Cricket_X	0.426	0.236	0.287	0.278	0.249	**0.203**
Cricket_Y	0.356	0.197	0.2	0.259	0.197	**0.156**
Cricket_Z	0.38	0.18	0.239	0.263	0.205	**0.156**
DiatomSizeRed.	0.065	0.065	0.101	0.126	0.065	**0.059**
ECGFiveDays	0.203	0.203	0.07	0.183	0.218	0.178
FaceAll	0.286	0.192	0.231	0.234	**0.144**	0.152
FaceFour	0.216	0.114	0.034	0.051	0.125	0.091
FacesUCR	0.231	0.088	0.109	0.09	0.102	**0.063**
fiftywords	0.369	0.242	0.277	0.209	0.226	**0.18**
fish	0.217	0.16	0.154	0.08	0.154	0.034
GunPoint	0.087	0.087	**0.047**	0.011	0.073	0.007
Haptics	0.63	0.588	0.565	**0.488**	0.571	0.584
InlineSkate	0.658	0.613	0.675	0.603	0.586	0.567
ItalyPowerDemand	0.045	0.045	**0.033**	0.096	0.044	0.039
Lightning2	0.246	0.131	0.18	0.257	0.131	**0.115**
Lightning7	0.425	0.288	0.263	0.262	0.26	**0.233**
MALLAT	0.086	0.086	0.072	0.037	0.075	0.05
MedicalImages	0.316	0.253	**0.232**	0.269	0.258	0.245
MoteStrain	0.121	0.134	0.118	0.135	0.205	0.114
NonInvasive.Thorax1	0.171	0.185	**0.103**	0.138	-	0.178
NonInvasive.Thorax2	0.12	0.129	**0.094**	0.13	-	0.112
OliveOil	0.133	0.167	0.1	**0.09**	0.167	0.133
OSULeaf	0.483	0.384	0.426	0.329	0.372	**0.194**
Plane	0.038	**0**	-	-	-	**0**
SonyAIBORobot.	0.141	0.141	0.235	0.175	0.185	0.293
SonyAIBORobot.II	0.305	0.305	0.177	0.196	0.123	0.124
StarLightCurves	0.151	0.095	0.036	**0.022**	0.066	0.079
SwedishLeaf	0.213	0.157	0.109	**0.075**	0.117	0.085
Symbols	0.1	0.062	0.121	**0.034**	0.059	0.049
SyntheticControl	0.12	0.017	0.023	**0.008**	0.027	0.01
Trace	0.24	0.01	**0**	0.02	0.01	0.01
TwoLeadECG	0.253	0.132	0.112	**0.046**	0.138	0.067
TwoPatterns	0.09	0.0015	0.053	0.001	0.004	**0**
UWaveGesture._X	0.261	0.227	0.213	**0.164**	0.211	0.199
UWaveGesture._Y	0.338	0.301	0.288	**0.249**	0.278	0.283
UWaveGesture_Z	0.35	0.322	0.267	**0.217**	0.293	0.29
wafer	0.005	0.005	0.047	0.004	0.006	0.003
WordSynonyms	0.382	0.252	0.381	0.302	0.243	0.226
yoga	0.17	0.155	0.157	0.149	0.156	**0.121**

Table 5.8: The full results of the state of the art in TSC on the UCR datasets (part two of two). The results are reported as error rates and split into two parts for readability. The first part is shown in Table 5.7, and part two includes results for Bag of Patterns (BoP) [72], Logical Shapelets (LS) and Fast Shapelets (FS) [107, 88], Fusion Methods for TSC (FUSION) [20], and the Transformation-Based Ensemble for TSC (TE) [3]. The results of the Proportional Elastic Ensemble (PROP) are repeated for comparison. Dashes indicate where results were not available, and the best result on each dataset across both tables is shown in bold.

Dataset	BoP	LS	FS	FUSION	TE	PROP
Adiac	0.432	0.414	0.514	0.352	0.358	0.353
Beef	0.433	0.433	0.447	-	0.4	0.367
Car	-	-	-	0.283	-	**0.167**
CBF	0.013	0.114	0.053	-	0.171	0.002
ChlorineConcentration	**0.036**	0.382	0.417	0.473	-	0.36
CinC_ECG_torso	-	0.301	0.174	**0.043**	-	0.062
Coffee	-	0.036	0.068	-	0.214	**0**
Cricket_X	-	-	-	-	-	**0.203**
Cricket_Y	-	-	-	-	-	**0.156**
Cricket_Z	-	-	-	-	-	**0.156**
DiatomSizeRed.	-	0.199	0.117	-	-	**0.059**
ECGFiveDays	-	0.006	0.004	-	-	0.178
FaceAll	0.219	0.341	0.411	-	0.281	0.152
FaceFour	**0.023**	0.511	0.09	0.116	0.148	0.091
FacesUCR	-	0.338	0.328	0.077	-	**0.063**
fiftywords	0.466	-	-	0.224	0.352	**0.18**
fish	0.074	0.223	0.197	-	0.194	0.034
GunPoint	0.093	0.107	0.061	-	0.053	0.007
Haptics	-	-	-	0.587	-	0.584
InlineSkate	-	-	-	**0.462**	-	0.567
ItalyPowerDemand	-	0.064	0.095	-	-	0.039
Lightning2	0.164	0.574	0.295	0.206	0.23	**0.115**
Lightning7	0.466	0.452	0.403	0.251	0.301	**0.233**
MALLAT	-	0.344	**0.033**	0.1	-	0.05
MedicalImages	-	0.413	0.433	0.234	-	0.245
MoteStrain	-	0.168	0.217	**0.07**	-	0.114
NonInvasive.Thorax1	-	-	-	-	-	0.178
NonInvasive.Thorax2	-	-	-	-	-	0.112
OliveOil	0.133	0.167	0.213	-	0.167	0.133
OSULeaf	0.256	0.314	0.359	0.228	0.417	**0.194**
Plane	-	-	-	-	-	**0**
SonyAIBORobot.	-	0.14	0.314	-	-	0.293
SonyAIBORobot.II	-	0.154	0.215	-	-	0.124
StarLightCurves	-	-	-	0.15	-	0.079
SwedishLeaf	0.198	0.187	0.269	0.115	0.157	0.085
Symbols	-	0.357	0.068	-	-	0.049
SyntheticControl	0.037	0.53	0.081	-	0.083	0.01
Trace	**0**	**0**	0.002	-	0.2	0.01
TwoLeadECG	-	0.144	0.09	-	-	0.067
TwoPatterns	0.129	0.461	0.113	-	0.165	**0**
UWaveGesture._X	-	-	-	-	-	0.199
UWaveGesture._Y	-	-	-	-	-	0.283
UWaveGesture_Z	-	-	-	-	-	0.29
wafer	0.003	**0.001**	0.004	-	0.002	0.003
WordSynonyms	-	-	-	**0.214**	-	0.226
yoga	0.17	0.26	0.249	0.401	0.163	**0.121**

losing on 2 (FaceFour and Trace). The main support for the BoP algorithm is that it is much faster than DTW, but these results indicate that the speed-up comes at the detriment of accuracy. The difference is significant.

- Fast shapelets (FS) and logical shapelets (LS) [107, 88] were evaluated on 33 UCR datasets. However, one result is ambiguous; it reported as Cricket, when there are actually 3 Cricket problems. Therefore the comparison is made over 31 datasets, and the proportional ensemble is more accurate than both shapelet tree algorithms on 26 of the 31 data sets, losing on 5. The difference is significant.
- The fusion technique (FUSION) proposed in [20] was evaluated on 20 UCR data sets. The elastic ensemble was more accurate on 14 of these, losing on 6. The result is significant.
- The elastic ensemble is also significantly better than the previous ensemble (TE) described [3], winning on 21 out of 25 common data sets. The difference is significant.
- Finally, an interval-based SVM classifier from [93] is evaluated on four UCR data sets (CBF, Two Patterns, Trace, and GunPoint). The results are not included in Table 5.7 or Table 5.8 due to lack of data, but the ensemble performed better on three of these simple problems. The fourth dataset, Trace, was only very slightly less accurate.

The results of the comparison of the EE to other work are very promising. Of the seven approaches with a representable number of published results, the EE significantly outperformed all seven on the UCR datasets that results were available for.

5.6 Conclusions

This chapter focused on time domain classification, which is the area of research in TSC that has generated most interest in the literature. We have addressed many aspects of TSC in the time domain throughout this chapter, ranging from reaffirming the commonly held benchmark in TSC, to proposing a new state-of-the-art classifier through information observed by running millions of individual experiments. A summary of the main contributions in this chapter are as follows:

- We demonstrated that NN classifiers with DTW are not outperformed by alternative classification approaches (Section 5.2), supporting the belief that NN classifiers

are hard to beat.

- We showed that setting the warping window for DTW 1-NN through cross-validation on training data (DTWCV) significantly improves the classifier (as first recognized in [90]). We also show that setting the number of neighbours in the NN classifier for TSC may be beneficial (Section 5.3.1), but the support for this is not as strong as setting measure parameters (Section 5.3.2). We recommend that setting measure parameters through CV should take priority over setting neighbourhood sizes, as it may be intractable to search for both parameters on large problems. We advocate the best configuration for approaching TSC problems with DTW is to use a 1-NN classifier, and to set the DTW warping window through cross-validation (DTWCV).

- No elastic measure that we tested was significantly more accurate than DTW when combined with a 1-NN classifier (Section 5.4), suggesting that DTW is still a worthy benchmark in TSC research.

- Combining the elastic measures using simple ensemble schemes produces a classifier that is significantly more accurate than any of the constituent parts over the full set of 75 datasets and the UCR datasets (Section 5.5.3). While the accuracies of the individual measures were not significantly different, the individual predictions made by each were significantly different (Section 5.5.1). To counter any suggestions of bias introduced through our selection of datasets, we repeated this experiment using the UCR datasets only, and the EE remained significantly more accurate than DTW with 1-NN. Comparisons to recent work in the literature (Section 5.5.4) lead us to believe that these are the best TSC results ever published, and the EE is the first classifier to significantly outperform DTWCV on the UCR datasets.

Chapter 6

Shapelet Domain Classification: The Shapelet Transform

Contributing Publications

- Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 289–297. ACM, 2012.
- Jason Lines and Anthony Bagnall. Alternative quality measures for time series shapelets. In *Intelligent Data Engineering and Automated Learning-IDEAL 2012*, pages 475–483. Springer Berlin Heidelberg, 2012.
- Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.

6.1 Introduction

The majority of TSC research has focused on similarity between series in the time-domain, typically building 1-NN classifiers with alternative distance measures (Chapter 5). Despite the evidence in favour of 1-NN approaches, various alternatives have been proposed (see Chapter 2 for a summary). Often these approaches are defined to allow the inclusion of more complex classifiers, such as decision trees or support vector machines, or to incorporate data transformations to evaluate time series similarity

in other data domains, such as similarity in change (Section 4.3) or global shape (Section 4.1).

A recent approach proposed transforming data to assess a previously under represented class of similarity: local shape-based similarity. *Time Series Shapelets* were proposed in [107] as subsequences from a dataset that are representative of class membership (for a full description, see Section 4.2). The original research proposed recursively extracting shapelets from a dataset to create simple decision tree classifiers. The best shapelet is extracted using information gain to assess the quality of candidates, and the data is partitioned according to a distance threshold. This process is repeated recursively until a full decision tree classifier is defined.

The key benefit of using shapelets for TSC is that they allow for phase-independent shape-based similarity to be detected between series, a type of similarity that is often very difficult (or impossible) to detect with algorithms that consider whole series. In addition, numerous secondary benefits are also provided by this approach, such as relatively fast classification times due to the compact nature of shapelets, and results achieved using shapelets being directly interpretable. This is because shapelets are subsequences from the actual data, so explanatory insight can be gained from observing how shapelets discriminate between series of different classes.

The aim of [107] was to use shapelets to create a classifier that was highly interpretable. This goal justified the selection of a decision tree implementation, which allows for clear explanatory analysis of the decision process. However, in general decision tree classifiers are often outperformed by other approaches and are typically more time consuming to build due to the recursive nature of the algorithm. This is further compounded in the case of shapelets, as the extraction process (described in Section 4.2) is relatively time consuming. The subroutine must be called many times, and each time shapelet quality must be evaluated using information gain. The original authors use information gain as it is interpretable and lends itself well to a decision tree approach, but they highlight that it is a bottle-neck in the running time of shapelet extraction. To counter this, they propose a novel entropy early abandon. However, this procedure is not defined for problems with more than two classes and does not generalise well to multi-class problems. In the worst of cases, the early abandon can actually have a negative impact on the speed of shapelet extraction.

In this chapter, we propose a novel algorithm to transform time series data into the shapelet domain. The key goal is to disassociate shapelet discovery from the classification process by removing the dependency on the decision tree implementation. By doing so, shapelet-transformed data can be combined with other classification algorithms to

maintain the interpretability provided by shapelets, while increasing classification accuracy and reducing training time. Furthermore we address the multi-class problem of shapelet discovery with IG by assessing an alternative quality measure, leading to a further study to investigate two additional alternative measures. The findings from these investigations culminates in the *shapelet transform*. The transformation we propose handles shapelets in three distinct stages. Firstly, the algorithm conducts a single scan of the data to extract the best k shapelets. Note that whilst k is a parameter to set, it is simply a cut-off value for the maximum number of shapelets to store and has no effect on the quality of the individual shapelets that are extracted. Secondly, we introduce a simple cross-validation approach for automatically setting shapelet length parameters for use in the transform, and investigate a similar method for automatically setting k too. Finally, a new transformed data set is created where each attribute represents a shapelet, and the value of each attribute is the distance from the shapelet to the original series. The key motivation for transforming the data in this way is that we can disassociate shapelet finding from building a classifier, allowing the transformed data set to be used in conjunction with any classifier. To summarise, the contributions in this chapter are as follows:

1. We introduce a novel caching algorithm to store the k best shapelets from a dataset in a single pass.
2. We propose and evaluate alternative shapelet quality measures for simpler multi-class implementation. These include using the F-stat from Analysis of Variance (ANOVA), Mood's Median, and Kruskal-Wallis.
3. We provide a parameter-free method for automatically setting parameters to extract significant shapelets from a dataset.
4. We evaluate shapelet-transformed data with various classification algorithms, and compare results against the original tree-based shapelet algorithm. We demonstrate that for the datasets we test with, using shapelet-transformed data with non-tree-based classifiers significantly outperforms the original tree-based shapelet algorithm.

6.2 Datasets

For the development stage of the shapelet transform, we selected 18 data sets from the UCR time series repository (see Section 3.1) and introduced 8 new datasets consisting of

hand outlines for classifying bone age (as introduced in Section 3.3). We selected these particular data sets because they have relatively few cases; even with optimisation, the shapelet algorithm is time consuming. Additional speed-ups and new datasets have been incorporated in the work of [52] where a refined shapelet transform is proposed, and the results are presented at the end of this chapter using over 70 datasets. However, these enhancements are not the contribution of this chapter. Therefore we report the majority of results using the original set of problems that were used for developing the transform, which we also used in the original publication of the algorithm in [78]. The datasets are listed in Table 6.1.

Table 6.1: The datasets that are used throughout the experiments with the shapelet transform.

Datasets			
Adiac	Beef	ChlorineConcentration	Coffee
DiatomSizeReduction	ECGFiveDays	ElectricDevices	FaceFour
GunPoint	ItalyPowerDemand	Lighting7	MedicalImages
MoteStrain	SonyAIBORobotSurface	Symbols	SyntheticControl
Trace	TwoLeadECG	DP_Little	DP_Middle
DP_Thumb	MP_Little	MP_Middle	PP_Little
	PP_Middle	PP_Thumb	

6.3 The Shapelet Transform

The following section outlines the shapelet transform that we propose. In Section 6.3.1 we introduce the algorithm for extracting shapelets from a dataset in a single iteration. In Section 6.3.2 we define the approach for transforming data using extracted shapelets. Then we focus on techniques for estimating the parameters required by the transform. In Section 6.3.3 we discuss setting the limit for the number of shapelets we store, k, and in Section 6.3.4 we introduce a simple cross-validation approach for estimating minimum and maximum shapelet length parameters.

6.3.1 Extracting the k Best Shapelets

The goal of the shapelet transform is to remove the dependency on the decision tree structure for shapelets by uncoupling shapelet discovery from the classifier. Therefore it becomes possible to search for shapelets in one stage only, removing the requirement to repeatedly find shapelets. The algorithm that we define for searching for shapelets within a dataset over a single pass is shown in Algorithm 7.

The algorithm begins by processing data in a very similar manner to the original

Algorithm 7 ShapeletCachedSelection($\mathbf{T}$, min, max, k)

```
1:  kShapelets = ∅;
2:  for all time series T_i in T do
3:     shapelets = ∅;
4:     for l = min to max do
5:        W_{i,l} = generateCandidates(T_i, l);
6:        for all subsequence S in W_{i,l} do
7:           D_S = findDistances(S,T);
8:           quality = assessCandidate(S, D_S);
9:           shapelets.add(S, quality);
10:    sortByQuality(shapelets);
11:    removeSelfSimilar(shapelets);
12:    kShapelets = merge(k, kShapelets, shapelets);
13: return kShapelets;
```

shapelet selection algorithm (Algorithm 6). For each series, all subsequences with lengths within the range of min and max are processed. However, rather than processing all candidates and only storing the best, the caching algorithm stores all candidates for the current series with an associated measure of quality (line 9). It should be noted that the original shapelet extraction algorithm used IG to assess shapelet quality. Unlike the shapelet decision tree however, the shapelet transform does not require an explicit split point to extract shapelets. Therefore many alternative measures can be considered for assessing shapelet quality, which is investigated in detail in Section 6.4. However for simplicity when defining the algorithm, the measure of similarity can simply be considered as IG to mirror the original algorithm.

Once all candidates for the series have been assessed, they are sorted in order of quality and self-similar shapelets are removed. We define two shapelets as being self-similar if they are taken from the same series and have overlapping indices. Once we have the set of non self-similar shapelets for a series, we merge these with the current best shapelets and retain the top k, and continue to iterate through the data until all series have been processed.

We store shapelets separately and merge later, rather than keeping the best k on-the-fly, to ensure that the best shapelets remain once self-similar candidates are pruned. For example, if we have a set of the best k shapelets at a given point in the algorithm, the next candidate that is processed could potentially overlap with one or more existing shapelets in the store. If this current shapelet is deemed to be better, all overlapping self-similar candidates must be removed from the store. A problem would then arise if

this new shapelet is improved upon later by a candidate that is deemed to be self-similar. This would force the recently added shapelet to be removed, and the previously removed shapelets would potentially become valid once more. However, as each subsequence is only processed once, it would be impossible to retrieve the previously removed candidates at this point even if they ultimately should be in the set of the best k shapelets. Therefore to avoid this problem, we initially extract all possible shapelets from a series, sort them by their quality and then remove those that are self similar in order. This gives priority to the best shapelets and ensures that no candidates are prematurely removed. We can then safely merge these with the existing k best shapelets before moving on to processes the next series.

6.3.2 Data Transformation

One of the main motivations of the proposed transformation is to allow shapelets to be used with a diverse range of classification algorithms. Rather than restricting them to classification through decision tree structures, our algorithm uses shapelets to transform instances of data into a number of features that can then be treated as a generic classification problem. The transformation process is defined in Algorithm 8.

Algorithm 8 ShapeletTransform(Shapelets S, Dataset D)

```
1: output = ∅;
2: for all time series ts in D do
3:    transformed = ∅;
4:    for all shapelets s in S do
5:       dist = subsequenceDist(ts, s);
6:       transformed.add(dist);
7:    output.add(transformed);
8: return output;
```

The transformation process is carried out using the subsequence distance calculation described in Section 4.2. Firstly, a set of k shapelets, S, is generated from the training data T, as seen in the previous section. For each instance of data T_i, the subsequence distance is computed between T_i and S_j, where $j = 1, 2, \ldots, k$. The resulting k distances are used to form a new instance of transformed data, where each attribute corresponds to the distance from each shapelet to the original time series. When using data split into training and test partitions, the shapelet extraction is carried out on only the training data to avoid bias; these shapelets are then used to transform each instance of the training and test data to create transformed data sets, which can then be used with any

traditional classification algorithm.

6.3.3 Setting k in the Shapelet Transform

The number of shapelets used in the transform is controlled through the use of the parameter k. While it should be noted that this is simply an upper bound on the number of shapelets that are used to transform data, the value of k can potentially have a large influence on classification results. Using too few shapelets would not provide enough information to make informed classification decisions, whilst using too many could overfit classifiers trained with the transformed data and dilute the influence of important shapelets. In the experiments with the shapelet transform, we use two strategies for selecting the number of shapelets to use in the filter for a given set of data; firstly, as a benchmark we use $\frac{n}{2}$ shapelets in the filter, where n is the number of attributes in a single series of the data. The second approach automatically selects the number of shapelets to use based on the results of a 5-fold cross-validation experiment.

This is performed by initially partitioning the training data into five equal parts. For each fold, we use the data as a testing set and combine the four other folds to form a set of training data. We then pass the training data into our filtering algorithm and produce p shapelets. These shapelets are used to create p different sets of transformed training data, where the first set is the original training data transformed by one shapelet, the second is transformed by two, and so on until the final set consists of p transformed features. This same procedure is applied to the testing fold, creating p sets of transformed test data. Given the class of the classifier that we wish to use the final shapelet with, we train a range of new classifiers using the p sets of transformed training data and classify the appropriate transformed test fold. Therefore, for each of the five folds we obtain p classification accuracies, each corresponding to the number of shapelets used to transform the data.

The value of p with the best overall accuracy across all five folds of the data is selected as the value of k for the number of shapelets that are used in the final filter. In cases where multiple values obtain the best results, we evaluated three strategies for selecting a single value from the set of best values: pick the smallest, median or largest value. We found that picking the largest marginally outperformed using the median value, whilst both approaches performed better than selecting the smallest value. Therefore for this approach, we always split ties using the largest value of p with the highest cross-validation accuracy.

6.3.4 Setting Shapelet Length Parameters

The original shapelet extraction algorithm in [107] required two length parameters to be set: *min* and *max*. The shapelet algorithm that we defined in Algorithm 7 also requires these two parameters. These values define the range of possible shapelet lengths, making the algorithm more efficient by reducing the search space. However, setting the parameters incorrectly can be detrimental to the outcome of the shapelet transformation if they prevent the most informative subsequences from being considered. To accommodate running the shapelet filter on a range of data sets without any specialised knowledge of the data, we define a simple algorithm for estimating the min and max parameters.

Algorithm 9 EstimateMinAndMax(T)

1: $shapelets = \emptyset$;
2: $length = T_1.length$;
3: **for** $i = 1$ to 10 **do**
4: $randomiseOrder(T)$;
5: $T' = [T_1, T_2, ..., T_{10}]$;
6: $currentShapelets = ShapeletCachedSelection(T'$, 1, $length$, 10);
7: $shapelets.add(currentShapelets)$;
8: $orderByLength(shapelets)$;
9: $min = shapelets_{25}.length$;
10: $max = shapelets_{75}.length$;
11: **return** min, max;

The procedure outlined in Algorithm 9 takes 10 random series from the dataset T and uses Algorithm 7 to find the 10 best shapelets within this small subset of data. The search parameters here are set from 1 to m, where m is the length of a whole series in T. Effectively this means that no constraint is placed on the length of possible shapelets. This is repeated 10 times in total, producing a set of 100 shapelets. The shapelets are sorted in order of length, and the lengths of the 25th and 75th shapelets are extracted and returned as *min* and *max* respectively. Note that this will not necessarily result in the optimal solution for parameter finding. However, it was important that we could adopt an automatic approach to approximate *min* and *max* parameters across a number of datasets to allow us to compare our filter fairly against the original tree implementation of shapelets, as the topic of shapelet length estimation was not covered in the original work of [107]. Therefore we use this approach to approximate *min* and *max* for each data set and build all shapelet trees and filters using these values throughout the experiments.

6.4 Alternative Shapelet Quality Measures

The original shapelet tree approach in [107] used information gain (IG) to assess the quality of shapelet candidates (see Section 4.2). The motivation for using this measure was two-fold. Firstly, IG is suitable for identifying how to produce a partition of the data, which is necessary when recursively dividing data to create a tree structure. Secondly, the authors proposed an optimistic early abandon technique for IG to make the measure more efficient by early pruning of poor candidates. However when implementing the shapelet transform, we considered alternative quality measures for shapelet discrimination for the following reasons:

1. The goal for the shapelet transform is to generate a set of shapelets from an entire data set, rather than specifically identifying how well a candidate splits the data. If we have the list of distances D_S, our goal is not to find the best way of partitioning D_S. Instead, we address the issue of how different are the lists $D_S^1, D_S^2, \ldots, D_S^c$, where D_S^j contains all the distances from the candidate to time series of class j.

2. The upper bounding technique for information gain relies on identifying the ideal partition of a number of unevaluated distances. As previously discussed, the utility of this approach degrades with multi-class problems as a simple binary split is impossible with more than two class values. In the most pessimistic case, all possible optimistic combinations of unevaluated distances must be considered. It is not hard to motivate how this may quickly become untenable, especially in the most extreme cases (for example, the Adiac dataset from Chapter 3 contains 37 distinct class values).

We believe that there are several alternative measures that we could adopt in place of IG for assessing the difference in distributions between class distances. The simplest approach would be to use the F-statistic from ANOVA (F-stat), and we also consider Kruskal-Wallis (KW) and Mood's Median (MM) as alternative quality measures (See Section 4.2 for more details). To test the utility of these alternative quality measures, we use IG, F-stat, KW and MM to create four shapelet tree classifiers according to the original implementation of [107]. We implement these measures within the context of shapelet trees, rather than the shapelet transform, to focus specifically on the effect of using alternative measures for judging shapelet quality without other implementation issues affecting the results. The test accuracies of using these classifiers are presented in Table 6.2. The KW implementation of the shapelet tree has the lowest average rank. However, the IG tree only records the best accuracy on 3 datasets, while F-stat and MM

win outright on 11 and 7 datasets respectively. A critical difference diagram is shown in Figure 6.1 to assess whether there is any significant difference between these classifiers.

Table 6.2: Test accuracies for variants of the shapelet-tree classifier, using information gain (IG), Kruskal-Wallis (KW), F-stat and Mood's Median (MM) as measures of shapelet quality.

Dataset	IG	KW	F-stat	MM
Adiac	**29.92%**	26.60%	15.60%	27.11%
Beef	50.00%	33.33%	**56.67%**	30.00%
ChlorineConcentration	**58.80%**	51.95%	53.52%	52.11%
Coffee	96.43%	85.71%	**100%**	85.71%
DiatomSizeReduction	72.22%	62.11%	**76.47%**	44.77%
DP_ Little	65.44%	68.00%	60.31%	**71.00%**
DP_ Middle	70.53%	69.33%	61.86%	**73.67%**
DP_ Thumb	58.11%	**72.00%**	55.97%	70.33%
ECGFiveDays	77.47%	87.22%	**99.00%**	92.80%
ElectricDevices	45.10%	44.16%	50.41%	**53.17%**
FaceFour	**84.09%**	44.32%	75.00%	40.91%
GunPoint	89.33%	94.00%	**95.33%**	92.00%
ItalyPowerDemand	89.21%	90.96%	**93.10%**	91.06%
Lighting7	**49.32%**	47.95%	41.10%	27.40%
MedicalImages	48.82%	47.11%	**50.79%**	48.95%
MoteStrain	82.51%	**83.95%**	**83.95%**	**83.95%**
MP_ Little	66.39%	69.67%	57.83%	**70.33%**
MP_ Middle	71.01%	**75.00%**	60.93%	72.00%
PP_ Little	59.64%	**72.00%**	58.60%	67.33%
PP_ Middle	61.42%	68.33%	58.14%	**69.67%**
PP_ Thumb	60.83%	71.33%	59.07%	**73.00%**
SonyAIBORobotSurface	84.53%	72.71%	**95.34%**	74.87%
Symbols	77.99%	55.68%	**80.10%**	57.39%
SyntheticControl	94.33%	90.00%	**95.67%**	85.67%
Trace	98.00%	94.00%	98.00%	**100%**
TwoLeadECG	85.07%	76.38%	**97.01%**	85.34%
Average Rank	2.5577	2.7885	2.3269	2.3269

There is no significant difference detected between the test accuracies of the four shapelet trees. However, accuracy is not the only method for evaluating these approaches. We have previously noted that IG is a time consuming calculation, even using the entropy early abandon introduced in [107]. While we consider test accuracy to be the most important evaluation criteria of a classification algorithm, some problem domains demand fast approaches for real-time scenarios, making the runtime of a classifier important too.

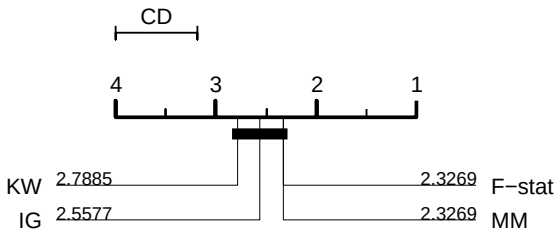


Figure 6.1: Critical difference diagram of the test accuracies from the four shapelet tree classifiers using alternative quality measures

With this in mind we conduct a timing experiment to compare the four measures, with the caveat that the IG early abandon is not implemented. We acknowledge that this may be due to an issue with our implementation, but justify our approach since we could apply the same early abandon used for IG to the other similarity measures. To create a direct comparison between the measures, our experiments compare the time taken by each measure to extract the best shapelet only from each dataset, rather than the time taken to build the complete tree. We believe that this is the fairest approach, since the measures will likely build trees of varying depths and this would influence the build times of the classifiers. We are interested in comparing similarity measures for shapelets for general use, such as being used in the shapelet transform, rather than focusing only on the shapelet transform. The timing results for finding the best shapelet with each measure are shown in Table 6.3

The timing results uncover a number of interesting facts. F-stat is clearly the fastest quality measure, winning on more datasets than the other three measures combined. F-stat is fastest for 19 of the problems, while IG is not fastest on any. In fact, IG is the slowest measure on all but 3 of the problems. Informally this result appears to be significant. To investigate this further, a critical difference diagram is shown in Figure 6.2 to test whether the timing ranks of the classifiers are significantly different.

The results demonstrate that using any of the three alternative quality measures for shapelet discovery is significantly faster than using IG. F-stat clearly has the lowest rank of the four and is also significantly faster than KW, but is still within the same clique as MM. Considering the results of these two preliminary studies using tree classification and timing experiments, there is good evidence to support considering alternative measures of shapelet quality in place of IG. The first result showed that no alternative measure

Table 6.3: Timing results for finding the best shapelet in each dataset using information gain (IG), Kruskal-Wallis (KW), F-stat and Mood's Median (MM)

Dataset	IG	KW	F-stat	MM
Adiac	17758.48	4974.5	**4509.91**	4752.63
Beef	1284.46	1253.17	1251.21	**1228.51**
ChlorineConcentration	26233.51	16699.23	**15681.39**	16572.67
Coffee	261.27	264.75	**258.15**	263.82
DiatomSizeReduction	55.35	54.61	**53.91**	54.36
DP_Little	97508.12	80556.13	**78005.7**	82052.11
DP_Middle	106382.47	94081.04	**91208.52**	91664.8
DP_Thumb	149567.07	125334.92	**123766.49**	124508.41
ECGFiveDays	151.64	150.9	**149.1**	155.43
FaceFour	4695.97	4621.97	**4556.41**	4648.45
GunPoint	592	569.51	**569.42**	580.76
ItalyPowerDemand	3.18	1.56	1.75	**1.46**
Lighting7	15497.07	15157.93	**14912.74**	14940.2
MedicalImages	15703.95	8148.76	**7742.97**	8111.36
MoteStrain	11.55	11.02	**10.76**	11
MP_Little	108518.67	89849.25	**88071.5**	89634.65
MP_Middle	156750.2	135852.37	**134731.54**	134756.02
PP_Little	97987.27	**79285.08**	79993.31	80514.6
PP_Middle	68730.32	59579.27	**57815.02**	58389.16
PP_Thumb	110204.43	**91183.51**	91401.49	91202.87
SonyAIBORobotSurface	7.8	6.79	**6.73**	6.79
Symbols	8992.15	8941.21	**8901.28**	8922.01
SyntheticControl	2280.82	1029.95	984.36	**974.27**
Trace	54829.06	55155.36	**54128.53**	54205.65
TwoLeadECG	3.61	3.15	3.12	**3.11**
Average Rank	3.84	2.7	1.32	2.14

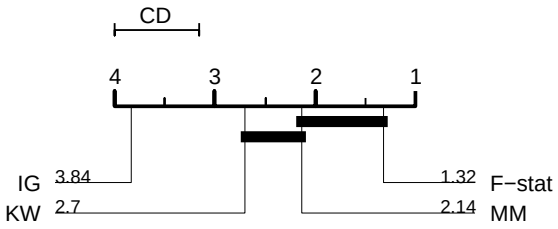


Figure 6.2: Critical difference diagram of the timings to find the best shapelet with the four alternative shapelet quality measures

produced a significantly worse classifier than IG, while the second result demonstrated all alternatives are faster for shapelet extraction than IG. We revisit alternative quality measures when implementing the shapelet transform in Section 6.5 to investigate whether these findings translate to our new approach.

6.5 Experimental Design

We have proposed several changes to the way shapelets can be used for classification and present a range of experiments to test these changes. For each, we use a simple train/test split and all reported results are testing accuracy. To ensure no bias is introduced into result, all shapelet selection, model selection, and classifier construction is done exclusively on the training set, and the test set is only used by the final trained classifier to report classification accuracy. All algorithms and experiments were implemented in Java within the Weka framework, and the shapelet transform is implemented as a Weka batch filter to allow easy integration into existing classification code.

6.6 Results

The experiments described in this section are designed to answer the following questions:

1. Does separating shapelet extraction from the classification model have a significant effect on accuracy? (Section 6.6.1)
2. Does the use of alternative shapelet quality measures affect accuracy or provide faster extraction? (Section 6.6.2)
3. Can shapelet-transformed data be combined with more complex classifiers to significantly outperform the original tree-based approach? (Section 6.6.3)
4. Is the method for automatically setting k in the transform robust, or is it advantageous to use a fixed constant for k? (Section 6.6.4)
5. How does the shapelet transform perform with the heterogeneous ensemble classifier defined in Section 2.5, and how does performance compare to DTW and the elastic ensemble over the extensive set of datasets used in Chapter 5? (Section 6.6.6)

6.6.1 Embedded Shapelets vs. Transformed Shapelets

Our first objective is to establish that separating shapelets from the classifier does not reduce classification accuracy. We implemented a shapelet decision tree classifier as

described in [107], and compared results to a C4.5 decision tree trained and tested on shapelet transformed data (using information gain as the quality measure and $\frac{n}{2}$ shapelets). Table 6.4 shows the results for the 26 data sets used.

Table 6.4: Shapelet tree classification vs. C4.5 classification with $\frac{n}{2}$ shapelet filtered features

Data	Shapelet Tree	C4.5
Adiac	29.92%	24.30%
Beef	50.00%	60.00%
ChlorineConcentration	58.80%	56.48%
Coffee	96.43%	85.71%
DiatomSizeReduction	72.22%	75.16%
ECGFiveDays	77.47%	96.17%
ElectricDevices	54.90%	53.45%
FaceFour	84.09%	76.14%
GunPoint	89.33%	90.67%
ItalyPowerDemand	89.21%	90.96%
Lighting7	49.32%	53.42%
MedicalImages	48.82%	44.87%
MoteStrain	82.51%	84.42%
SonyAIBORobotSurface	84.53%	84.53%
Symbols	77.99%	47.14%
SyntheticControl	94.33%	90.33%
Trace	98.00%	98.00%
TwoLeadECG	85.07%	85.25%
DP_Little	65.44%	65.92%
DP_Middle	70.53%	71.24%
DP_Thumb	58.11%	57.99%
MP_Little	66.39%	63.43%
MP_Middle	71.01%	73.25%
PP_Little	59.64%	57.40%
PP_Middle	61.42%	62.49%
PP_Thumb	60.83%	59.53%

The shapelet tree was best on 13 datasets, C4.5 with transformed data on 12, and they were tied on one. There is no significant difference detected by a paired t-test or a Wilcoxon signed rank test. Therefore this experiment provides no evidence that performing shapelet extraction prior to constructing the decision tree makes the classifier less accurate. This is a crucial result as it motivates further study to investigate using shapelet-transformed data with alternative classification approaches.

6.6.2 Using F-stat with the Shapelet Transform

The results in Section 6.6.1 have shown separating shapelet extraction from the classification algorithm does not affect the accuracy of classifiers built. Therefore the shapelet transform is clearly worthy of further study with classifiers other than the C4.5. However, the results of investigating alternative measures of shapelet quality in Section 6.4 raise one final question about the implementation of the transform: should we use IG, or should we consider alternative measures of shapelet quality? To investigate this question, we create a shapelet transform using the most promising alternative measure from the results in Section 6.4: F-stat.

We earlier demonstrated that using F-stat in the tree approach was significantly faster than IG and provided test classification results that were not significantly different from the IG approach. Therefore we form a simple experiment to investigate whether this result holds when using the shapelet transform. We reuse the results of the C4.5 classifier with shapelet transformed data using IG from Section 6.6.1 and compare against a C4.5 classifier trained on data that was transformed by shapelets extracted with F-stat. The results are reported in Table 6.5.

The results show that over the 26 datasets, the IG transform wins on 11 datasets, F-stat on 12, and they tie on 3. There is no significant difference between the ranks of the two classifiers. Therefore we can conclude that there is good potential for using F-stat, and possibly other alternative measures, in place of IG. We have already demonstrated that F-stat is significantly faster for shapelet extraction than IG, and have now demonstrated that there is no significant difference in classification performance of C4.5 classifiers trained using shapelets extracted with either measure of quality.

However, for the duration of this chapter we continue to use IG in the shapelet transform. We justify this decision because qualitatively at least, we feel that IG may produce more discriminatory shapelets. Through the added information of an explicit split point, shapelets extracted with IG are likely to relate to how a single class differs from the population, providing extra insight into the classification decision that F-stat will likely not provide. Furthermore the main aim of the experiments is to discover whether we can significantly improve classification accuracy with shapelets by separating the extraction process from the classifier. By using an alternative quality measure, we may risk clouding any conclusions that we draw from results on shapelet transformed data. Therefore by continuing to use the IG implementation of the shapelet transform, we can fairly and directly compare our results to the original tree-based approach towards shapelets, attributing and gains in classification accuracy to the shapelet transform,

Table 6.5: The accuracies of C4.5 decision tree classifiers built on shapelet-transformed data using information gain (IG) and F-stat as shapelet quality measures. Positive values in the difference column indicate better performance with IG, and negative values represent better performance by F-stat.

Dataset	**IG**	F-**Stat**	**Difference**
Adiac	24.30%	16.88%	7.42%
Beef	60.00%	63.33%	-3.33%
ChlorineConcentration	56.48%	53.75%	2.73%
Coffee	85.71%	85.71%	0.00%
DiatomSizeReduction	75.16%	73.20%	1.96%
ECGFiveDays	96.17%	98.14%	-1.97%
ElectricDevices	53.45%	50.41%	3.04%
FaceFour	76.14%	62.50%	13.64%
GunPoint	90.67%	93.33%	-2.66%
ItalyPowerDemand	90.96%	90.96%	0.00%
Lighting7	53.43%	53.43%	0.00%
MedicalImages	44.87%	41.58%	3.29%
MoteStrain	84.43%	86.82%	-2.39%
SonyAIBORobotSurface	84.53%	94.51%	-9.98%
Symbols	47.14%	52.86%	-5.72%
SyntheticControl	90.33%	79.67%	10.66%
Trace	98.00%	97.00%	1.00%
TwoLeadECG	85.25%	93.68%	-8.43%
DP_Little	65.92%	67.69%	-1.77%
DP_Middle	71.24%	70.41%	0.83%
DP_Thumb	57.99%	59.88%	-1.89%
MP_Little	63.43%	63.20%	0.23%
MP_Middle	73.25%	66.04%	7.21%
PP_Little	57.40%	58.34%	-0.94%
PP_Middle	62.49%	65.44%	-2.95%
PP_Thumb	59.53%	65.44%	-5.91%
Average	69.55%	69.39%	0.16%

rather than the measure of shapelet quality that we use.

6.6.3 Alternative Classifiers with Shapelet-transformed Data

The main motivation of the shapelet transform is to uncouple shapelet extraction from the classifier in order to investigate whether using more complex, non-tree-based classifiers can significantly improve classification accuracy. To test this, the set of 26 datasets were transformed using the shapelet filter, with $k = \frac{n}{2}$ and shapelet lengths estimated using the simple cross-validation procedure from Section 6.3.4. Seven standard classification algorithms were implemented with default Weka parameters in this experiment along with the original shapelet tree algorithm: C4.5, 1-NN, Naïve Bayes, Bayesian Network, Random Forest, Rotation Forest, and linear Support Vector Machine (See Section 2.4 for more details). The test accuracies for these classifiers are shown in Table 6.6.

The linear SVM is the best performing classifier with an average rank of 2.87, and is the best overall classifier on 10 out of the 26 problems. Informally, it is clear that the average rank of the SVM with shapelet-transformed data is much lower than the shapelet tree. Figure 6.3 shows a critical difference diagram of the results in Table 6.6.

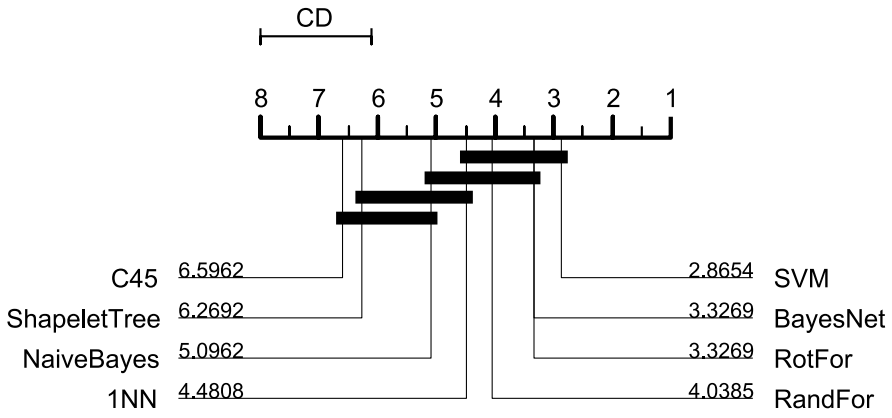


Figure 6.3: Critical difference plot for eight shapelet-based classifiers derived from the results in Table 6.6

There is a clear division in performance between the majority of the simpler classifiers (shapelet tree, C4.5, and Naïve Bayes) and the more complex classifiers. Rotation forest, Random Forest, Bayesian Networks and SVM are all within a higher clique than the shapelet tree, demonstrating that the ranks of these classifiers are significantly better than the shapelet tree. While there may be a trade-off between interpretability and accuracy with this approach (for example, it is easier to understand the decision process

Table 6.6: Testing accuracy and ranks of 8 classifiers constructed on the shapelet transform with $n/2$ shapelets. The table is broken into two parts for readability, but rank calculations are valid across both tables.

Data	Shapelet Tree		C4.5		1-NN		Naïve Bayes	
Adiac	29.92%	(3)	24.30%	(7)	25.32%	(5)	28.13%	(4)
Beef	50.00%	(8)	60.00%	(6.5)	83.33%	(3)	73.33%	(4)
ChlorineConcen.	58.80%	(2)	56.48%	(6)	56.93%	(5)	45.96%	(8)
Coffee	96.43%	(4.5)	85.71%	(8)	**100.00%**	**(1.3)**	92.86%	(6)
DiatomSizeRed.	72.22%	(8)	75.16%	(7)	**93.46%**	**(1)**	78.76%	(6)
ECGFiveDays	77.47%	(8)	96.17%	(6)	98.37%	(4)	96.40%	(5)
ElectricDevices	54.90%	(2)	53.45%	(4)	24.25%	(7)	25.37%	(5)
FaceFour	84.09%	(7)	76.14%	(8)	**100.00%**	**(1.5)**	97.73%	(4.5)
GunPoint	89.33%	(8)	90.67%	(7)	98.00%	(4)	92.00%	(6)
ItalyPowerDemand	89.21%	(8)	90.96%	(7)	92.13%	(4.5)	92.52%	(2)
Lighting7	49.32%	(7.5)	53.42%	(6)	49.32%	(7.5)	57.53%	(5)
MedicalImages	48.82%	(4)	44.87%	(6)	45.66%	(5)	17.37%	(8)
MoteStrain	82.51%	(8)	84.42%	(7)	**90.34%**	**(1)**	88.82%	(3)
SonyAIBORobot.	84.53%	(6)	84.53%	(5)	84.03%	(7)	79.03%	(8)
Symbols	77.99%	(6)	47.14%	(8)	85.63%	(2)	77.99%	(7)
SyntheticControl	**94.33%**	**(1)**	90.33%	(4)	93.00%	(2)	78.00%	(7)
Trace	98.00%	(5)	98.00%	(5)	98.00%	(5)	98.00%	(5)
TwoLeadECG	85.07%	(8)	85.25%	(7)	**99.47%**	**(1)**	99.12%	(3)
DP_Little	65.44%	(8)	65.92%	(7)	72.78%	(6)	73.49%	(3)
DP_Middle	70.53%	(8)	71.24%	(7)	73.73%	(6)	73.96%	(5)
DP_Thumb	58.11%	(7)	57.99%	(8)	60.71%	(6)	62.96%	(5)
MP_Little	66.39%	(7)	63.43%	(8)	68.52%	(6)	68.76%	(5)
MP_Middle	71.01%	(7)	73.25%	(4)	70.89%	(8)	71.95%	(5)
PP_Little	59.64%	(7)	57.40%	(8)	67.22%	(5)	69.23%	(4)
PP_Middle	61.42%	(8)	62.49%	(7)	68.52%	(6)	69.82%	(5)
PP_Thumb	60.83%	(7)	59.53%	(8)	67.69%	(6)	69.35%	(4)
AverageRank	6.27		6.60		4.48		5.10	

Data	BayesianNet		RandForest		RotForest		SVM (linear)	
Adiac	25.06%	(6)	30.43%	(2)	**30.69%**	**(1)**	23.79%	(8)
Beef	**90.00%**	**(1)**	60.00%	(6.5)	70.00%	(5)	86.67%	(2)
ChlorineConcen.	57.08%	(4)	57.58%	(3)	**63.52%**	**(1)**	56.15%	(7)
Coffee	96.43%	(4.5)	**100.00%**	**(1.3)**	89.29%	(7)	**100.00%**	**(1.3)**
DiatomSizeRed.	90.20%	(3)	80.39%	(5)	83.01%	(4)	92.16%	(2)
ECGFiveDays	**99.54%**	**(1)**	93.26%	(7)	98.61%	(3)	98.95%	(2)
ElectricDevices	53.63%	(3)	**55.98%**	**(1)**	24.25%	(7)	24.25%	(7)
FaceFour	**100.00%**	**(1.5)**	87.50%	(6)	98.86%	(3)	97.73%	(4.5)
GunPoint	99.33%	(2)	96.00%	(5)	98.67%	(3)	**100.00%**	**(1)**
ItalyPowerDemand	92.42%	(3)	**93.00%**	**(1)**	92.03%	(6)	92.13%	(4.5)
Lighting7	65.75%	(2.5)	64.38%	(4)	65.75%	(2.5)	**69.86%**	**(1)**
MedicalImages	28.16%	(7)	50.79%	(3)	51.45%	(2)	**52.50%**	**(1)**
MoteStrain	89.06%	(2)	84.58%	(6)	86.98%	(5)	88.66%	(4)
SonyAIBORobot.	**89.68%**	**(1)**	85.19%	(4)	89.02%	(2)	86.69%	(3)
Symbols	**92.26%**	**(1)**	84.62%	(3.5)	84.42%	(5)	84.62%	(3.5)
SyntheticControl	76.67%	(8)	89.00%	(5)	92.00%	(3)	87.33%	(6)
Trace	**100.00%**	**(1)**	98.00%	(5)	98.00%	(5)	98.00%	(5)
TwoLeadECG	98.77%	(4)	96.14%	(6)	97.98%	(5)	99.30%	(2)
DP_Little	72.90%	(5)	73.02%	(4)	74.67%	(2)	**75.15%**	**(1)**
DP_Middle	74.67%	(4)	75.50%	(3)	76.80%	(2)	**79.64%**	**(1)**
DP_Thumb	63.91%	(4)	64.14%	(3)	67.10%	(2)	**69.82%**	**(1)**
MP_Little	69.47%	(4)	71.36%	(3)	**75.15%**	**(1)**	75.03%	(2)
MP_Middle	71.12%	(6)	75.15%	(2)	74.67%	(3)	**76.92%**	**(1)**
PP_Little	70.06%	(2)	66.63%	(6)	69.82%	(3)	**72.07%**	**(1)**
PP_Middle	71.36%	(3)	70.53%	(4)	75.38%	(2)	**75.86%**	**(1)**
PP_Thumb	69.47%	(3)	67.81%	(5)	72.78%	(2)	**75.50%**	**(1)**
Average Rank	3.33		4.04		3.33		2.87	

of a tree compared to a SVM), the main conclusion is that by separating the shapelet discovery from the classifier, there is greater potential for accurate solutions while still retaining the insight provided from discovered shapelets. Interestingly, while not significantly outperforming the original shapelet tree, nearest neighbour classification once again demonstrates strong performance for TSC as 1-NN also falls into the top clique.

6.6.4 Shapelet Selection

The results in Section 6.6.3 for the alternative classifiers with shapelet-transformed data were achieved by setting $k = \frac{n}{2}$ when extracting shapelets. We proposed an alternative method for setting k in Section 6.3.3 through cross-validation, which if used could make the transform effectively parameter-free when coupled with the length parameter estimation described in Section 6.3.4. To determine whether this approach is robust, we calculated the number of shapelets to use for each dataset and repeated the classification experiments from Table 6.6 using the alternative classifiers. The results are reported in Table 6.7.

The results show that using an automatically selected number of shapelets set through cross-validation improves the average classification accuracy of 7 out of the 8 classifiers. In all cases, the classifiers achieve equal or better results on over half of the data sets, with the Random Forest classifier improving the most overall.

6.6.5 Exploratory Data Analysis

The results in Section 6.6.3 show that using shapelets to transform data can provide promising classification results. However, one of the strengths of using shapelets is that they allow a level of interpretation that other approaches do not. The original work on shapelets in [107] demonstrated this with examples where they document the decision trees that are built for datasets with the associated shapelets.

One of the key motivations behind the transform is to produce accurate and reliable decisions that are interpretable. To demonstrate that the transform retains the interpretability of the original shapelet algorithm, the GunPoint data is used as an example. The GunPoint data contains readings from an actor carrying out the motion of drawing a gun, and the classification problem is to determine whether or not they were holding a prop or just miming the action (the *Gun/NoGun* problem). In [107], the authors identified that the most important shapelet for classification was when the actor lowered their arm; if they had no gun, a phenomenon called overshoot occurred and caused a dip in the motion data. This is shown using the original figure from [107] in Figure 6.4.

Table 6.7: Relative accuracies of the classifiers when trained with an automatically selected number of shapelets through cross-validation vs. using $n/2$ shapelets (as in Table 6.6)

Data	C4.5	1-NN	NaïveBayes	BayesNet	RandForest	RotForest	SVM (linear)
Adiac	2.30%	0.00%	3.58%	5.37%	4.86%	7.16%	7.42%
Beef	-10.00%	-13.33%	0.00%	-6.67%	20.00%	0.00%	-3.33%
ChlorineConcentration	-1.09%	-2.63%	4.32%	0.36%	0.05%	0.83%	0.03%
Coffee	0.00%	0.00%	3.57%	0.00%	0.00%	3.57%	0.00%
DiatomSizeReduction	0.00%	-0.65%	-21.24%	-2.94%	1.96%	-1.63%	1.96%
ECGFiveDays	0.00%	-1.63%	1.74%	0.00%	5.92%	0.46%	-0.46%
ElectricDevices	0.00%	0.00%	0.00%	1.01%	0.37%	0.00%	0.00%
FaceFour	0.00%	0.00%	2.27%	0.00%	6.82%	-9.09%	1.14%
GunPoint	0.00%	0.00%	0.67%	0.00%	2.00%	0.00%	0.00%
ItalyPowerDemand	0.00%	3.89%	-3.89%	0.10%	0.68%	2.92%	3.21%
Lighting7	0.00%	-2.74%	2.74%	9.59%	-4.11%	2.74%	1.37%
MedicalImages	0.00%	3.03%	33.16%	23.29%	-0.79%	0.00%	3.16%
MoteStrain	0.00%	-0.08%	0.48%	-3.83%	1.12%	0.00%	0.56%
SonyAIBORobotSurface	0.00%	0.17%	0.00%	0.00%	-0.50%	0.00%	0.00%
Symbols	2.41%	-2.21%	-4.32%	1.31%	-2.71%	0.70%	-8.74%
SyntheticControl	-1.67%	0.67%	0.33%	0.33%	-2.00%	0.00%	3.33%
Trace	0.00%	0.00%	0.00%	-2.00%	0.00%	2.00%	0.00%
TwoLeadECG	0.00%	0.00%	-0.44%	0.70%	-3.16%	-1.76%	0.26%
DP_Little	-1.54%	1.54%	2.25%	2.37%	-2.49%	0.59%	1.42%
DP_Middle	1.07%	-1.89%	-1.07%	-0.83%	1.42%	0.83%	-0.12%
DP_Thumb	4.26%	0.36%	-0.95%	-1.42%	0.83%	0.59%	0.95%
MP_Little	1.18%	0.12%	1.89%	0.59%	0.36%	-1.54%	-1.78%
MP_Middle	1.42%	1.30%	-0.12%	0.47%	0.36%	3.43%	-0.12%
PP_Little	8.17%	0.83%	0.95%	0.59%	0.95%	-2.84%	-1.66%
PP_Middle	1.30%	1.18%	1.42%	1.42%	3.79%	-0.24%	-2.37%
PP_Thumb	1.18%	0.71%	1.30%	0.36%	-0.36%	-3.43%	0.36%
Average Improvement	0.35%	-0.44%	1.10%	1.16%	1.36%	0.20%	0.25%
Data Sets Improved	9	11	15	15	16	12	13

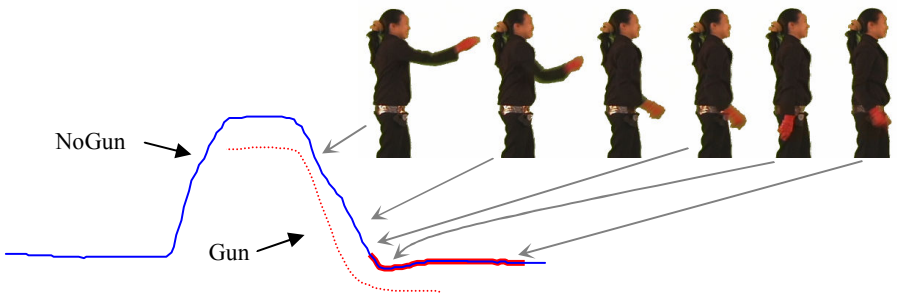


Figure 6.4: An illustration of the *Gun/NoGun* problem taken from [107]. The shapelet that they extract is highlighted at the end of the series.

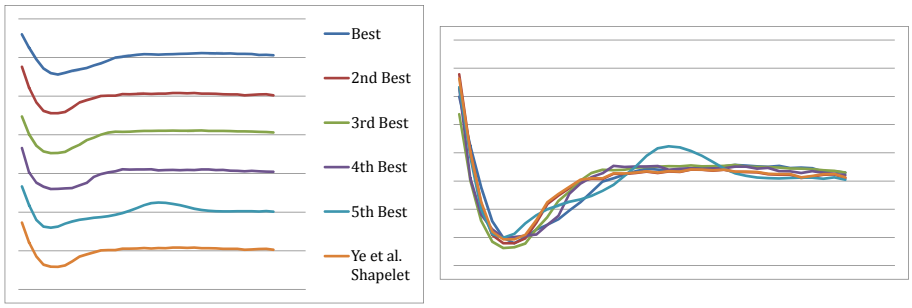


Figure 6.5: An illustration of the five best shapelets extracted by the shapelet transform. The graph to the right shows how closely matched they are when place on top of one another.

The tree originally built for the GunPoint data contains a single shapelet that corresponds to the arm being lowered back into position at the end of the series. To demonstrate that the shapelet filter retains this level of interpretability while extracting important information from the data, the transform was configured for the GunPoint data using the length parameters specified in the original paper to allow for a fair comparison between the two methods. The top five shapelets that were extracted by the shapelet transform are shown in Figure 6.5 along side the shapelet from Figure 6.4.

The top 5 shapelets are all closely matched with the original shapelet, reinforcing the claim that the shapelet transform produces interpretable results. Furthermore, if the top 10 shapelets are extracted, even further insight can be achieved. Figure 6.6 shows that the top ten shapelets extracted through the transformation process form two distinct clusters; the shapelets to the left correspond to the moments where the arm is lifted, and are also instances where there is a gun present in the actor's hand. It is intuitive

that these shapelets are important, since if the presence/absence of the gun when being returned to the holster is important, it should also be important when being removed from the holster. This is new insight that is not captured by the original shapelet tree algorithm.

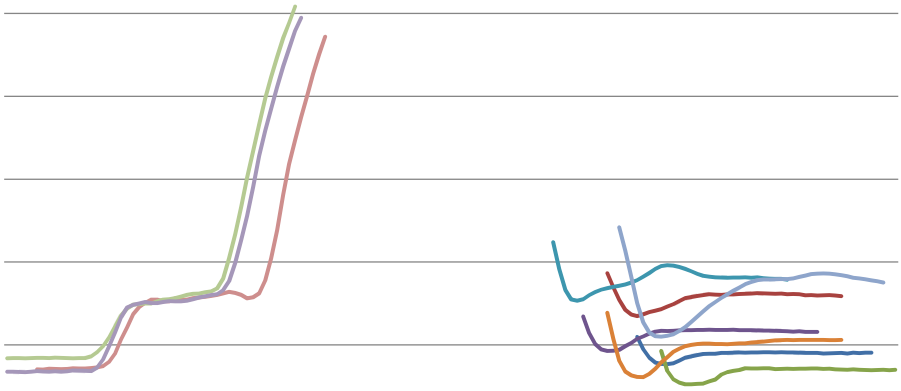


Figure 6.6: The 10 best shapelets for the *Gun/NoGun* problem extracted by the shapelet transform. The shapelets form two distinct clusters, the first where the arm is raised, and the second when the arm is lowered.

6.6.6 Comparison to Alternative Approaches

We have shown that through using the shapelet transform, classification with shapelets can be significantly improved by disassociating shapelet discovery from the classification algorithm to allow more complex and non-tree-based approaches to be used. However, these experiments do not offer any comparison of the shapelet transform to alternative TSC approaches. For example, while we would not expect a shapelet-based approach to outperform benchmarks such as 1-NN DTW across all TSC problems, we would expect the shapelet transform to provide better classification performance on problems where discriminatory features are embedded within local shapes. Additionally, since we introduced the shapelet transform into the literature in [78], two variants of the original shapelet approach have been proposed: logical shapelets [80] and fast shapelets [88]. Logical shapelets consider conjunctions and disjunctions of shapelets when constructing the tree-based classifier rather than single shapelets, and the fast shapelets variant provides an approach analogous to the original shapelet implementation, though implemented with various speed-up techniques to accelerate shapelet extraction.

To reaffirm the effectiveness of the shapelet transform, we present updated results from experiments carried out in [52] to compare the shapelet transform to alternative

approaches. Firstly, in light of the new shapelet-based approaches, we wish to report an updated comparison between the shapelet transform and the new approaches. The experimental support for logical and fast shapelets [80, 88] is provided through the use of 29 common UCR datasets. To create a fair comparison, we extended our original experiments in [52] to consider the same 29 datasets. Additionally, these datasets form a subset of the problems used in Chapter 5 when proposing the elastic ensemble (EE). This allows the opportunity to compare the transform and two new shapelet approaches to the EE and 1-NN DTW with warping set through cross-validation (DTWCV) too. The results of the shapelet transform using a linear SVM against these alternatives are reported in the critical difference diagram in Figure 6.7. The results demonstrate that

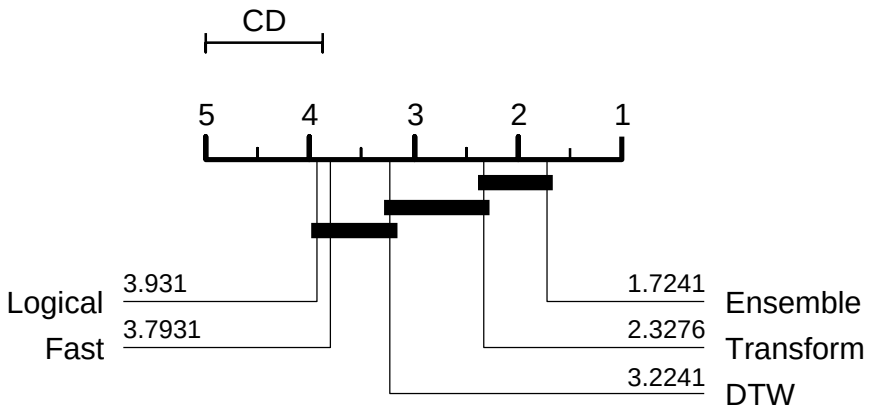


Figure 6.7: The average ranks for three shapelet algorithms, DTWCV and Elastic Ensemble on 29 UCR datasets. The critical difference for $\alpha = 5\%$ is 1.1137.

while the shapelet transform does not significantly outperform DTWCV, it is significantly more accurate than both logical and fast shapelets over the 29 datasets. This result suggests that the best current technique for using shapelets (in terms of accuracy at least) is to use them to transform data, rather than embedding shapelets within a decision tree classifier.

Additionally, the shapelet transform is not significantly outperformed by EE over these problems, lending further support to the use of the transform for TSC. However, comparing the shapelet transform to the EE over 29 datasets is restrictive when considering that we have previously provided results with the EE over 75 datasets in Chapter 5. Furthermore the shapelet transform used in this comparison only considered results using a linear SVM with shapelet-transformed data. Rather than limiting the transform to a single classification algorithm, we could use the heterogeneous ensemble that was de-

fined in the time domain in Section 2.5, using shapelet-transformed data rather than the raw time series. Using additional results with the shapelet transform that were recorded in [51], we can compare a heterogeneous shapelet ensemble to the EE over 71 datasets (shapelet ensemble results were not available for 4 datasets). The resulting critical difference diagram, including DTW and Euclidean distance 1-NN as benchmarks, is shown in Figure 6.8.

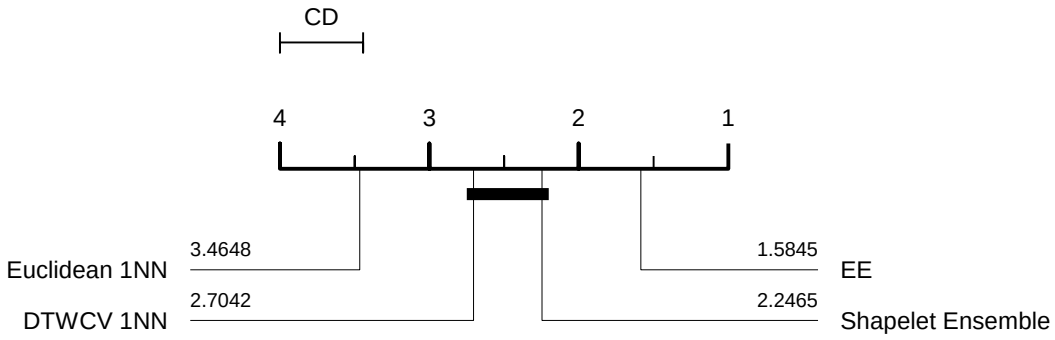


Figure 6.8: A critical difference diagram to compare a heterogeneous ensemble built with shapelet-transformed data to the EE, DTW 1-NN and Euclidean 1-NN over 71 datasets. The results show

The results show that over a larger and more diverse set of data, the EE is significantly better than the shapelet ensemble. However, the shapelet ensemble is not significantly outperformed by DTW and has a lower rank overall. This reaffirms the mantra of DTW being '*hard to beat*', but also underlines the effectiveness of the shapelet transform for TSC. As concluded previously in Chapter 5, Euclidean 1-NN is not a good benchmark for TSC algorithms to be judged against, as it is easily outperformed by all alternatives in this comparison.

6.7 Conclusions

The work in this chapter has proposed a shapelet transform for TSC that extracts the k best shapelets from a dataset in a single pass. We implement this transform using a novel caching algorithm to store shapelets, and apply a simple, parameter-free cross-validation approach for extracting the most significant shapelets. A total of 26 datasets are used

to demonstrate the effectiveness of the transform, and after an analysis of alternative shapelet quality measures that can be implemented within the transform, it is shown that a C4.5 decision tree classifier trained with shapelet-transformed data is competitive with an implementation of the original shapelet decision tree from [107]. Leading on from this, it is demonstrated that the transformed data can be applied to non-tree based classifiers to achieve significantly improved classification performance over the original shapelet tree, while still maintaining the interpretability offered by the shapelets. We demonstrate this through an in-depth case study of the *GunPoint* dataset, providing exploratory analysis of the extracted shapelets to demonstrate that the output is not only consistent with the original shapelet tree, but also offers even further insight into the problem. Finally, we produce an analysis of two extensions of the shapelet tree approach, and conclude that using shapelets to transform data is currently the best way to use shapelets by demonstrating that doing so leads to significantly better accuracy than any of the shapelet tree implementations, and when combined with a heterogeneous ensemble, the resulting shapelet ensemble offers competitive performance to DTW over a very large set of varied datasets.

Chapter 7

The Collective of Transformation-based Ensembles

In Chapter 5 an elastic ensemble (EE) was proposed for solving classification problems in the time-domain by combining the predictions of 1-NN classifiers built with various elastic distance measures. The results of the EE were, to the best of our knowledge, the first ever to outperform DTW 1-NN on the UCR datasets. However, we have previously argued in Chapter 4 that not all discriminatory features are detectable in the time domain. Similarity between series may be more easily detected in other domains, such as similarity in the frequency domain (Section 4.1) and similarity in change (Section 4.3). Additionally, Chapter 6 proposed a new time series transformation technique for uncovering local shape-based similarity: the shapelet transform. By transforming series into the shapelet domain, it was shown that complex classifiers could be applied to the transformed data to take advantage of the local shape-based similarity that can be detected with shapelets, while not being restrained by the limitations of using a decision tree classifier.

The goal of this chapter is to create a new classification algorithm that combines each of these four types of similarity into a single output. This is done through creating ensemble classifiers in the time, frequency, change, and shapelet domains, and combining them to form a collective of transformation-based ensembles (COTE). The EE proposed in Chapter 5 is used for classification in the time domain, and heterogeneous ensembles of various classification algorithms are formed for the remaining three domains (as defined in Section 2.5). Using 72 datasets, including the full set of 46 widely-used UCR datasets, the COTE is used to test the hypothesis that constructing a collective of ensemble classifiers built on different data representations improves mean classification accuracy

compared against classifiers built on single data representation. The contributions of this chapter are as follows:

1. Our primary contribution is to demonstrate that the simple collective formed by including all classifiers in one ensemble is significantly more accurate than any single constituent part, and is significantly more accurate than all published TSC algorithms evaluated on the train/test splits of UCR datasets that we know of. In addition to the 46 UCR datasets, we use a further 26 datasets in our evaluation, including a case study of the *Caenorhabditis elegans* problem introduced in Section 3.5.

2. Our secondary contribution is to investigate alternative hierarchical ensemble structures. The strategy of placing all classifiers in a single ensemble gives very accurate classifiers, but offers little exploratory insight into a particular problem. We investigate alternative hierarchical collective structures that use weighting schemes and selection schemes between ensembles on different transforms. We demonstrate that although most approaches give significantly worse accuracy than the flat approach of a single ensemble, a collective of transform-based ensembles where inclusion is determined by a Mann-Whitney rank sign test is not significantly worse.

The remainder of this Chapter is structured as follows. Firstly, the four individual transformation based ensembles are outlined and motivated through simple train/test experiments. The two novel classification approaches introduced in this thesis, the EE and shapelet transform, are analysed and motivated with a comparison to the current state-of-the-art in TSC. Following the establishment of the four ensembles, a simple flat ensemble strategy is defined and tested, leading on to an investigation into more intuitive and complex ensembling approaches. Finally a thorough comparison of the COTE to the leading TSC approaches from the literature is carried out to demonstrate that the Flat-COTE is the new benchmark for TSC on the UCR datasets.

7.1 Datasets

In Chapter 5 and Chapter 6 we originally used 75 datasets. However, the hand classification problems outlined were restructured as described in 3.3 to form the second generation hand-outline problems. For example, *DP_Little*, *DP_Middle*, and *DP_Thumb* were concatenated and replaced by *DistPhalanxOutline* by the author of the original project that generated these problems [30]. Therefore the datasets were updated to reflect the most recently available data for this problem domain. Additionally, the results

Table 7.1: Datasets grouped by problem type

Image Outline Classification					
DistPhalanxAge	DistPhalanxOutline	DistPhalanxTW	FaceAll	FaceFour	WordSynonyms
MidPhalanxAge	MidPhalanxOutline	MidPhalanxTW	OSULeaf	Phalanges	yoga
ProxPhalanxAge	ProxPhalanxOutline	ProxPhalanxTW	Herring	SwedishLeaf	MedicalImages
Symbols	Adiac	ArrowHead	BeetleFly	BirdChicken	DiatomSize
	FacesUCR	fiftywords	fish		
Motion Classification					
CricketX	CricketY	CricketZ	UWaveX	UWaveY	UWaveZ
GunPoint	Haptics	InlineSkate	ToeSeg1	ToeSeg2	MutantWorms2
		MutantWorms5			
Sensor Reading Classification					
Beef	Car	Chlorine	Coffee	Computers	SmallKitchen
FordA	FordB	ItalyPower	LargeKitchen	Lightning2	Lightning7
StarLightCurves	Trace	wafer	RefrigerationDevices	MoteStrain	Earthquakes
ElectricDevices	SonyRobot1	SonyRobot2	OliveOil	Plane	Screen
Human Sensor Reading Classification					
	TwoLeadECG	ECGFiveDays	ECGThorax1	ECGThorax2	
Simulated Classification Problems					
	MALLAT	CBF	SyntheticControl	TwoPatterns	

of two datasets with the shapelet transform were not available at the time of analysis. Therefore, due to the slight changes in available data, we list the full set of problems used in this chapter in Table 7.1. There are 72 datasets in total, and where comparisons are made to the earlier work of the EE and shapelet transform, we do so using the new set of 72 datasets to ensure that results are directly comparable. Additionally, it should be noted that this set of data still contains the full set of 46 UCR datasets [65], enabling results to be compared to other algorithms in the literature.

7.2 Transformation-based Ensembles

The COTE is formed by combining transformation-based ensembles built on four distinct data spaces. We use two specific types of ensemble for this; for the shapelet, change, and frequency domains we use a heterogeneous ensemble, as originally described in Section 2.5.4. For the time domain we use the elastic ensemble (EE) defined in Chapter 5. The specification of the heterogeneous ensemble is briefly revisited below for clarity, along with justification of using the EE in place of it for the time domain.

7.2.1 Heterogeneous Ensemble

To build heterogeneous ensembles in the shapelet, change, and frequency domains, we must first transform data into the correct representations. We do this using the approaches described in Chapter 4 and Chapter 6. For the frequency domain, we transform

data using the power spectrum as outlined in Section 4.1. For the change domain, we transform data using the three autocorrelation-based approaches that were discussed in Section 4.3, and concatenate ACF, PACF, and AR model terms to create output series. We choose to use all of the three possible approaches to keep the implementation as simple as possible. We could discriminate between the three possible transforms and select only one, but by providing terms for each of the three approaches, we provide classifiers with as much explanatory data in the change domain as possible. The intuition behind this is to keep the transform clear, and allow the classification algorithms to establish the best discriminatory features from all of the features that are available.

As outlined in Section 2.5, the classifiers that we include in the heterogeneous ensemble are the WEKA [48] implementations of:

- k Nearest Neighbour (where k is set through cross validation);
- Naïve Bayes;
- Bayesian Network;
- C4.5 decision tree [86];
- Support Vector Machine [28] with linear kernel;
- Support Vector Machine with quadratic kernel;
- Random Forest [16] (with 100 trees);
- Rotation Forest [94] (with 10 trees);

The ensemble is formed by adding the individual classifiers to a pool to form a democratic voting system. As discussed in Section 2.5, there are many ensembling schemes that can be used for weighting votes within the ensemble; in this instance, we choose to assign weights proportionally to cross-validation accuracy on the training data. We select the proportional scheme as it was the approach that performed best when developing the EE in Section 5.5.2, and including all four ensemble schemes that we originally considered in four data domains would make the implementation of the transformation-based ensemble more complex and less intuitive.

The set of classifiers used in the heterogeneous ensemble were chosen to balance both simple and complex classifiers, but the selection is fairly arbitrary and other classification algorithms could also be considered. Furthermore, no attempt to optimise parameter settings are made for these these classifiers through cross validation (with the exception of finding k for the k-NN classifier, which is done internally via Weka when the classifier is built). We chose to do this to reduce the complexity of the algorithm and to keep

the focus of this research on the importance of transformation in TSC. Nevertheless, a wider range of more sophisticated classifiers may improve performance and is worthy of further investigation.

7.2.2 Time Domain Classification with the Elastic Ensemble

It is intuitive to use the EE in place of the heterogeneous approach for the time domain since it was developed for that data space in particular, and the results in Section 5.5 demonstrated that the EE was significantly more accurate than DTW and other leading competition over a large range of data. To further justify this decision however, a simple experiment was carried out using the EE and a time domain heterogeneous ensemble on the datasets used in this chapter. Each were trained using the training data only, and the results of the test data accuracies are shown in scatter plot in Figure 7.1. To summarise, of the 72 datasets in use, the EE won on 46, lost on 23, and tied on 3. The difference in the average ranks of the classifiers was significant. After achieving these results, we also considered using the EE for the three remaining data domains. However, the results were less conclusive for change, frequency, and shapelet-transformed data, possibly due to the elastic measures originally being designed for use in the time domain, rather than transformed data spaces. We note that there is potential to investigate the EE for these domains further in the future, but for the work with COTE we restrict use of the EE to the time domain only and continue to use the heterogeneous ensemble in the other domains.

7.3 Results Using a Single Ensemble: Flat-COTE

The first configuration of the COTE that we propose is the simplest to implement; it is a logical extension of the EE in that it is simply a single ensemble consisting of all constituent classifiers from the EE in the time domain, combined with the constituents of the heterogeneous ensembles from the change, frequency, and shapelet domains (33 classifiers in total). We call this classifier the flat-COTE and it is implemented with a proportional voting scheme. While simple to comprehend, the implementation of the flat-COTE comes at a trade-off for interpretability, as it would be desirable to have a meta-ensemble that selects the correct constituent ensemble to use *a priori* when processing a dataset. Such a process would highlight where similarity can be best assessed for a given dataset. However, the justification for the flat-COTE is that if similarity exists in only one of the time, change, frequency, or shapelet domains, the proportional weighting of

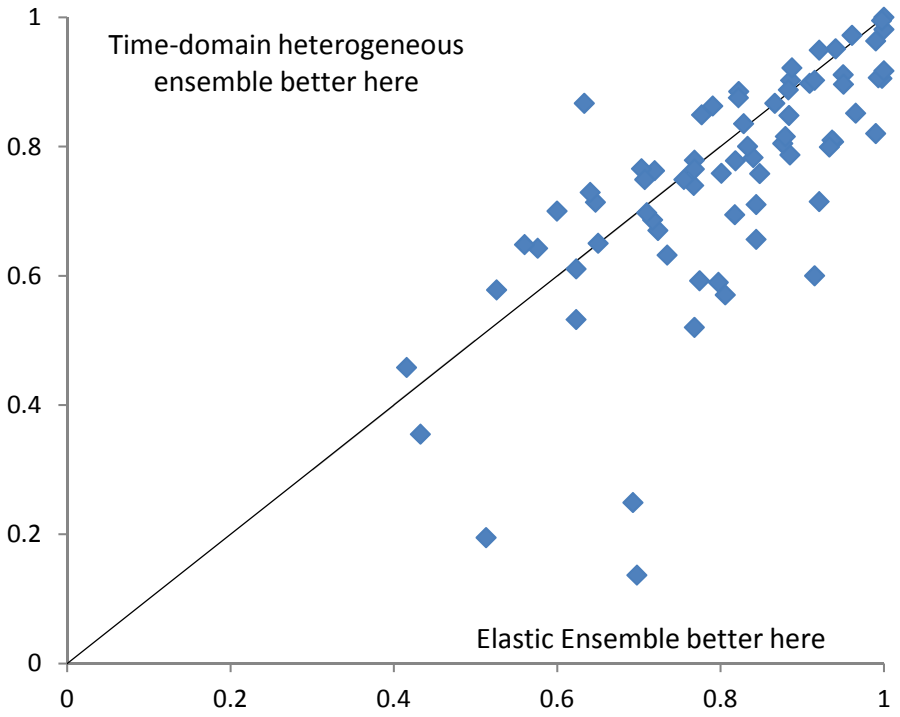


Figure 7.1: Test accuracy of the Elastic Ensemble defined in Chapter 5 and a heterogeneous ensemble built in the time domain over 72 problems.

the votes in that domain should overpower the remaining classifiers in the flat-COTE. In contrast, if similarity is detectable in multiple domains, the decision process should be informed by a more diverse set of representations than in the EE, potentially leading to better classification results for problems where similarity between series is not easily detected in the time domain. The test classification performance of the flat-COTE is compared to each of the four individual ensembles, and standard benchmarks in the form of Euclidean 1-NN and DTW 1-NN with warping set through cross-validation, in the critical difference diagram presented in Figure 7.2.

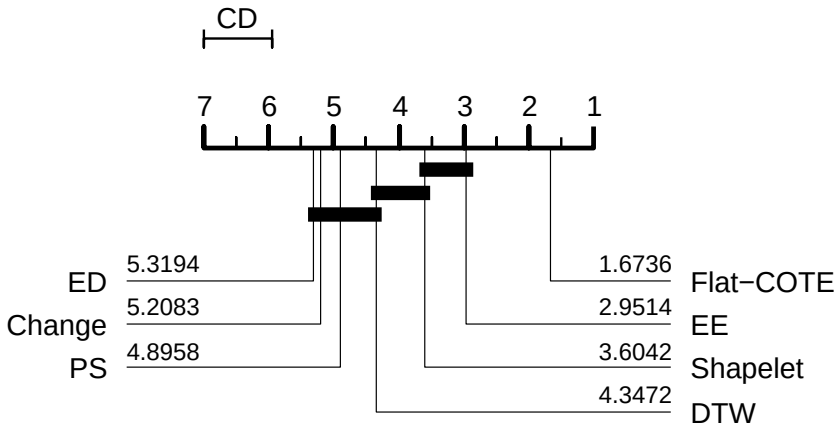


Figure 7.2: Critical difference diagram for collective (flat-COTE) and the individual ensembles for the change domain (Change), the power spectrum (PS), shapelet transform (Shapelet) and the time domain elastic ensemble (EE). Single classifiers 1-Nearest neighbour with Euclidean distance (ED) and dynamic time warping distance with warping window set through cross validation (DTW) are included for contrast.

The results in Chapter 5 demonstrated that the EE significantly outperformed the commonly-held benchmark of DTW 1-NN and various other elastic measures. Despite this finding, the results summarised in Figure 7.2 show that the flat-COTE is significantly more accurate than the EE over the 72 datasets. The information provided by the three transform ensembles within the flat-COTE therefore must discover discriminatory features that are hard (or perhaps impossible) to detect in the time domain.

These very positive results lead to two immediate questions. Firstly, how does the performance of the flat-COTE compare to other leading TSC approaches from the literature? Secondly, is it possible to improve the flat-COTE by selecting which transformation-based ensembles to include/exclude into the ensemble for a given problem? These two ques-

tions are answered throughout the remainder of this chapter following a case study to investigate how the flat-COTE performs on a problem that is not well suited to the time domain.

7.4 Case Study: Classifying Caenorhabditis elegans

The results of the flat-COTE have demonstrated that the new classifier is significantly more accurate than the EE over the 72 datasets. Since the EE is believed to be the first classifier to significantly outperform DTW 1-NN on the UCR datasets, it is interesting to explore how the flat-COTE has improved upon the EE. Specifically, as with the majority of research in TSC, the EE focuses on time domain similarity between series. However many problems exist where the search for similarity between data would be much better suited to other domains. In this case study, we explore two such datasets that are derived from the *Caenorhabditis elegans* introduced in Chapter 3 to explore how the flat-COTE improves upon the EE on problems that are not suited to the time domain. The test error rates of the flat-COTE and the four individual transformation-based ensembles on the two derived datasets (Worms2 and Worms5) are shown in Table 7.2.

Table 7.2: Test classification errors for the two-class and five-class worm problems with five different ensembles.

Dataset	Flat-COTE	EE	Shapelet	PS	Change
Worms5	0.25	0.38	0.30	0.26	0.19
Worms2	0.18	0.38	0.23	0.19	0.14

As expected, we observe that the time domain classifier is the worst of all for the worm datasets. This is unsurprising given the nature of the data, but underlines the merits of investigating similarity in domains other than time. Though flat-COTE outperforms the EE, it should be noted that it is less accurate than the best approach overall (Change domain). This emphasises that it is desirable to be able to determine the best transform *a priori*, so unsuitable representations (i.e. the time domain in this case) can be omitted from the ensemble to avoid adding noise to the predictions. The shapelet ensemble is less accurate than the power spectrum and change ensembles, but shapelets offer the added bonus of greater explanatory power. To emphasise this, Figure 7.3 shows the best shapelets for Worms2, the two-class problem (wild-type and mutant).

The best wild-type shapelet represents highly regular movement where the worm cyclically adopts the eigenworm1 shape. The mutant shapelet is much more erratic, with short localised variation from the regular pattern, which likely explains why the

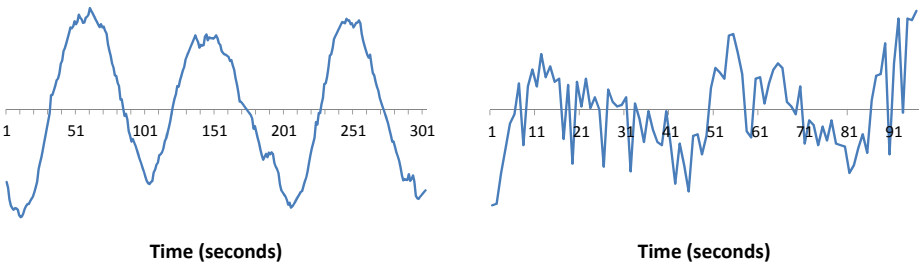


Figure 7.3: The best shapelets for wild-type (left) and mutant worms (right) for the two class problem.

Change transform is the best approach for this problem. The low order ACF terms for the non-mutants will be highly significant since the movement from one time step is highly correlated with the previous step, whereas this correlation will be much weaker for the mutant type. Further insight can be gleaned from observing the best shapelets extracted on the Worms5 dataset, as shown in Figure 7.4. We see the same localised variance with the mutants as with the two class problem, but there is also some variability in the degree of deviation from the eigenworm between mutants. This preliminary study has demonstrated that time series classification could provide a useful way of automating what is currently a very labour intensive process, and that the COTE approach gives very promising results and insights that the EE could not offer.

7.5 Comparison to Other Approaches

In the style of the comparison between the EE and other approaches in Section 5.5.4, we compare the flat-COTE to other recent work in TSC through pair-wise comparisons. This is achieved though testing for significant results by using the binomial test (BT) and Wilcoxon signed-rank test. However, unlike the original comparisons in Section 5.5.4, we also provide a further layer of analysis by comparing many of the recent approaches in a single critical difference diagram. We have not implemented the majority of these alternative approaches, hence this critical difference diagram is built predominantly using published results from the original work. This approach obviously limits the number of datasets that we can consider in the comparison as we have no control over which datasets are used. In spite of this however, by removing one approach with few common datasets from the critical difference diagram, we can compare the approaches over a set of 37 common datasets (out of a possible 46 UCR problems). The pair-wise comparisons of

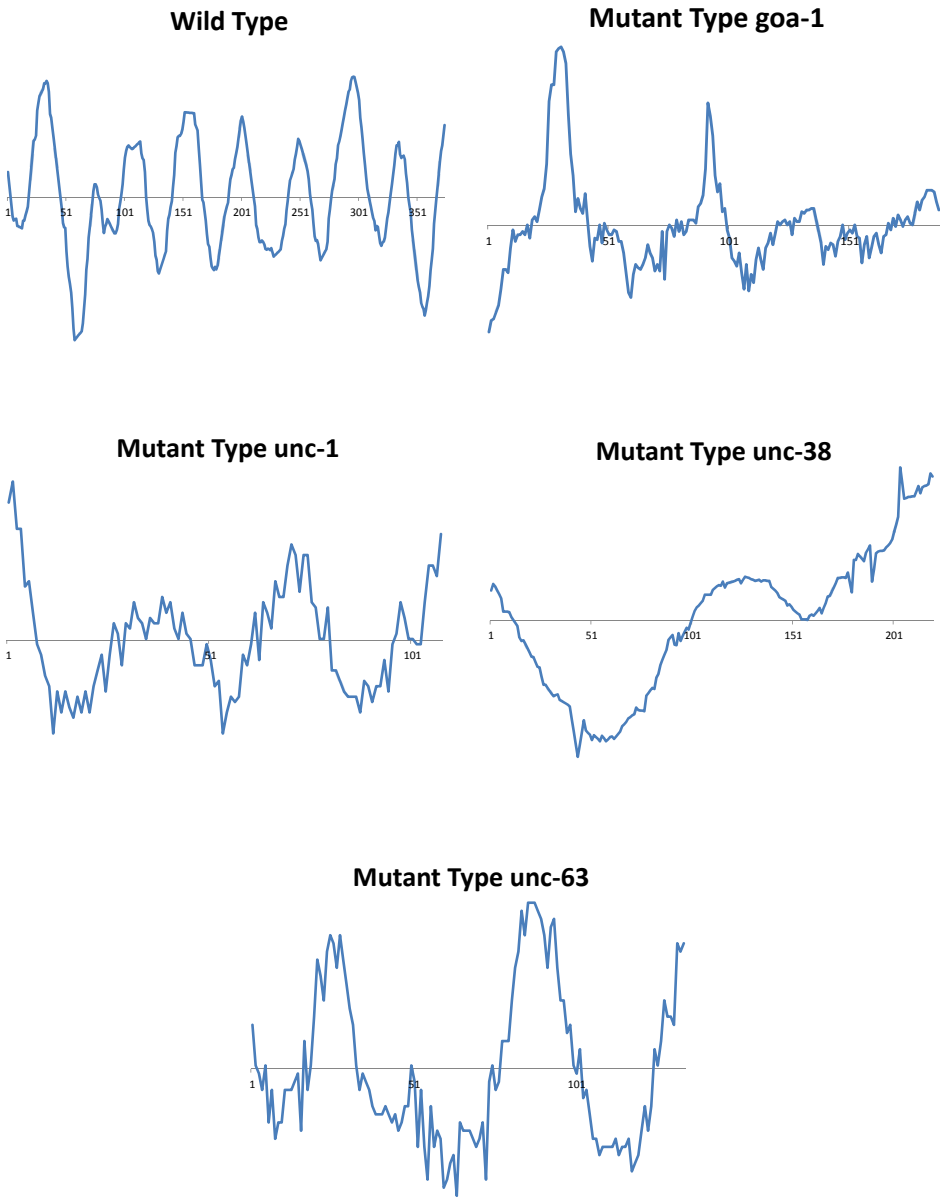


Figure 7.4: The best shapelets for the five class problem. It is interesting to note the regular pattern in the best shapelet for the wild type class, while the patterns exuded in the best shapelets for the mutant types appear to be much more erratic.

the flat-COTE to alternatives from the literature are as follows, and the critical difference diagram is shown in Figure 7.5

1. The Time Series Bag-of-Features (TSBF) classifier [11] is evaluated on 44 UCR datasets. In comparison to the best version of TSBF (TSBF Rand), flat-Cote wins on 37 datasets and loses on 7. Flat-COTE is significantly better at the 1% level. The p-values are 2.65×10^{-6} (BT) and 8.86×10^{-6} (WSR).

2. Two versions of Time Series Forest (TSF) [34], TSF entrance and TSF entropy, are assessed on 44 UCR datasets. In comparison to TSF entrance (the best version of TSF), flat-Cote wins on 35 datasets and loses on 9. Flat-COTE is significantly better at the 1% level. The p-values are 5.3×10^{-5} (BT) and 1.65×10^{-5} (WSR).

3. The complexity invariant distance (CID) [9] is evaluated on 42 UCR datasets (the two Fetal ECG datasets are missing). Flat-COTE is more accurate on 41 and worse on 1. Flat-COTE is significantly better at the 1% level. The p-values are 9.78×10^{-12} (BT) and 1.12×10^{-8} (WSR).

4. The recurrence patterns compression distance (RPCD) [99] reports test accuracy on 37 datasets (omitting simulated problem sets from the UCR data and the two fetal ECG datasets). Flat-Cote wins on 32 datasets, ties on 1, and loses on 4. Flat-COTE is significantly better at the 1% level. The p-values are 9.71×10^{-7} (BT) and 2.3×10^{-6} (WSR).

5. The Bag-of-Patterns (BOP) approach [72] is evaluated on 19 UCR datasets. Flat-COTE is better on 16 of these, ties on 1, and is worse on 2. Flat-COTE is significantly better at the 1% level. The p-values are 0.0007 (BT) and 0.002 (WSR).

6. Time Warp Edit (TWE) distance [79] with optimised parameters is evaluated on 19 UCR datasets. Flat-COTE is better on 16 of these, ties on 1, and is worse on 2. Flat-COTE is significantly better at the 1% level on these 19 datasets. The p values are 0.0007 (BT) and 0.002 (WSR). It is also significantly better on all 72 datasets (using the results of the internal TWE classifier within the EE of the flat-COTE).

7. The Move-Split-Merge distance metric (MSM) [101] is evaluated on 19 UCR datasets. Flat-COTE is better on 16 of these, ties on 1, and is worse on 2. Flat-COTE is significantly better at the 1% level. The p values 0.0007 (BT) and 0.002 (WSR). It is also significantly better on all 72 datasets (using the results of the internal MSM classifier within the EE of the flat-COTE).

8. DTW has been used as the standard benchmark algorithm for UCR datasets in the vast majority of TSC research. For the 46 UCR problems flat-COTE is better on 40 of these, ties on 3, and is worse on 3. Over all 72 datasets, flat-COTE is better on 63, worse on 6 and draws on 3.

9. Euclidean distance is also still often used to support new TSC algorithms. Flat-COTE is better on 44 of these, ties on 1, and is worse on 1. Over all 72 datasets, flat-COTE is better on 69, ties on 1 and is very marginally worse on 2.

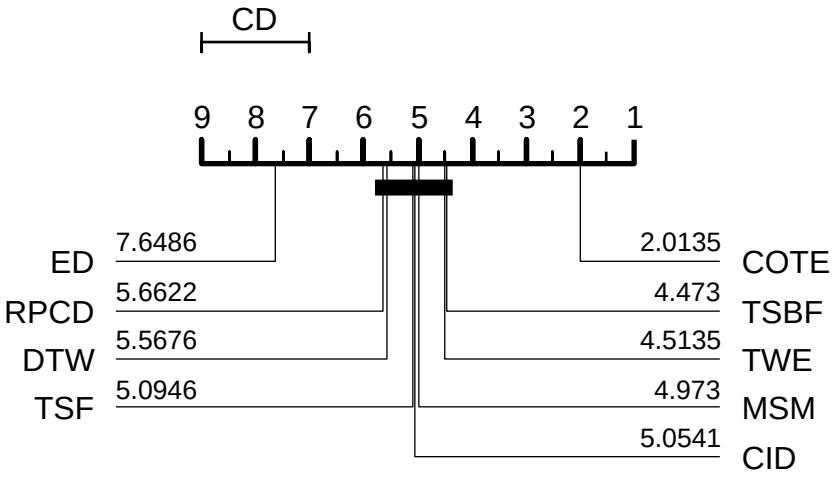


Figure 7.5: Critical difference diagram for flat-COTE and the other TSC algorithms on 37 UCR datasets that are common across each paper. BoP is omitted due to only having results for 19 datasets available, and the results for TWE and MSM were taken from [75] as the original work of [79] and [101] also only report results on 19 datasets.

The results of the pair-wise comparisons between flat-COTE and the other approaches provides evidence to support the use of flat-COTE, demonstrating that it is significantly more accurate than each of the alternatives we test against. This is reinforced in the critical difference diagram in Figure 7.5, which demonstrates that the flat-COTE has a significantly lower average rank across the 37 common datasets than any other algorithm that we test against. To the best of our knowledge, these are the best results ever

published on the UCR data. However, even with this in mind, when proposing new data mining algorithms it is also important to address limitations as well as strengths. There are numerous metrics that can be used for assessing the performance of new algorithms for TSC, and while we believe that accuracy is the most important for classification, it is also relevant to consider the time complexities of new approaches.

Due to flat-COTE being an ensemble of approaches, informally it is clear that it will be more time consuming than a simple nearest neighbour algorithm (such as DTWCV). Since flat-COTE is a combination of distinct processes that can run in parallel, the time complexity of the classifier will be equal to that of the most time consuming part: the shapelet transform. As discussed by [52], the shapelet search is enumerative. For a set of n series of length m, there are $n(m-l+1)$ possible shapelets of length l in each series. To assess a candidate shapelet, each must be compared to every series in the dataset with $O(m)$ distance calls, each requiring $O(l)$ point-wise calculations. This gives the time complexity for assessing a single shapelet as $O(nml)$, resulting in $O(n^2m^4)$ complexity for processing all shapelets in an entire dataset. To address this, much research into shapelets has focused on speeding up the shapelet discovery process, and such speed-ups could also be employed within flat-COTE. For example, the original shapelet work [107] proposed a simple early abandon during shapelet discovery. [88] propose online normalisation of subsequences to speed up discovery, and reorder candidate shapelets using summary statistics to enhance the potential for early abandonment. [74] propose alternative measures of shapelet quality to speed up shapelet discrimination for multi-class problems. [80] pre-calculate statistics between series based on cumulative sums and cross products to accelerate shapelet discovery by trading off space complexity in favour of better time complexity. However, we maintain our belief that the most important criteria for assessing new TSC algorithms is classification accuracy. Therefore we do not significantly alter the shapelet transform approach of [52] by implementing such speed-ups to ensure that the main contributions of this work are not clouded. Our main focus is to produce a classifier that significantly outperforms other TSC algorithms on the UCR datasets, and the results achieved in the section demonstrate that this has been successful. This leads us to our final question; now that we have an effective classifier, is it possible to improve the flat-COTE by implementing a scheme to automatically select the best, or a subset, of transformation-based ensembles that are most appropriate for a classification problem?

7.6 Alternative Ensemble Designs

The flat-COTE has been shown to produce very good results. In the initial comparison between the flat-COTE and EE in Section 5.5, we demonstrated that adding classifiers from the three additional transform domains (change, frequency and shapelet) to classifiers in the time domain produced a significantly better ensemble classifier than using only the time domain ensemble (i.e. the EE). However, the structure of the flat-COTE does not provide information into why the decision process is improved through adding the new classifiers. One way that this level of intuition could be provided is if we were to form a collection of independent ensembles in each transform domain, and then use a decision process to select (or weight) particular ensembles in the final collective ensemble. The result of this decision process alone would provide extra information into the transforms, highlighting the transformation domains where similarity is best detected for a dataset. To investigate this process, we propose three simple approaches that could be adopted within alternative COTE classifiers:

1. Predict the best outright transform to use during training, and omit the remaining three ensembles from the test classifier.

2. Weight each transformation-based ensemble to create a second layer of proportional voting.

3. Select a subset of transforms to use for a dataset by predicting the best, and then pruning any transforms that are significantly outperformed on the training data.

7.6.1 Best Internal Ensemble

The most intuitive of the strategies that we propose is to simply identify the best transformation space for a problem on the training data, and then use the appropriate ensemble in the testing phase. To motivate this strategy, Table 7.3 shows the distribution of the best transform across the 72 datasets judged solely on test accuracy

Table 7.3: Frequency of test set wins by transform

Transform	Number of datasets
Elastic Ensemble	34
Shapelets	22
Power Spectrum	9
Change	7

The data in Table 7.3 is retrospective, recorded from simply observing the test classification results using all four ensembles and then picking the best transform space after the fact. The goal of this COTE strategy is to predict this information in advance, with the hope that if this can be done accurately, the strategy can improve upon the flat-COTE by only using the classifiers that are deemed relevant to the problem in hand. However, surprisingly, this is not the case even if we could predict the best ensemble in advance with 100% accuracy we cannot outperform flat-COTE. The scatter plot of test accuracies in Figure 7.6 shows the flat-COTE versus a hypothetical oracle-COTE over the 72 datasets, where the accuracy is always provided from the single best constituent ensemble within the COTE.

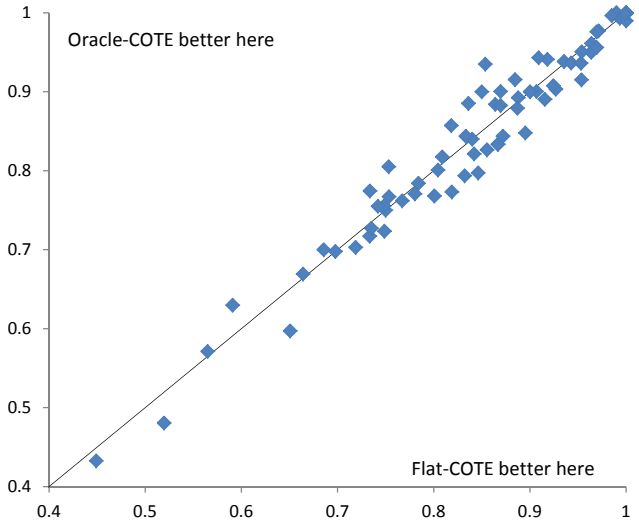


Figure 7.6: Test accuracy of oracle-COTE vs flat-COTE over 72 problems. Oracle-COTE wins on 27 datasets, flat-COTE wins on 36, and they tie on 9.

Even when predicting the best transform with 100% accuracy, the oracle-COTE is not significantly better than flat-COTE. Oracle-COTE wins on 27 dataset, flat-COTE on 36 and they tie on 9. This shows that there is clearly little to choose between the two ensembling schemes, even when assuming the most optimistic case for the oracle-COTE. This suggests that the power of the flat-COTE for TSC is achieved by considering features from multiple transform domains concurrently. However, despite this, it is still worth considering which ensemble performs best in a single domain since it proves greater explanatory power.

The most obvious information that we have available for attempting to identify the best transform in advance is the training cross-validation results. However, we found

that using the simple decision rule of picking the ensemble with the highest average training accuracy among constituent classifiers only identified the best transform for the test data correctly 55% of the time. We repeated this experiment with other summary statistics, such as median and max, but these faired no better. All of the strategies that we tested for picking the best transform on training accuracies resulted in a classifier that was significantly less accurate than the flat-COTE.

An alternative strategy that we also trialled was considering problem types in the context of a transfer learning problem. For example, if shapelets offer the best transformation domain for detecting similarity for one image outline problem, it would seem intuitive that shapelets would provide the best solution for another image outline problem. To quantify whether there appears to be such a relationship between datasets and transform domains, Table 7.4 shows the proportion of each dataset type won by the different transforms. The numbers are small so we cannot infer too much, but it appears that (perhaps as expected) motion problems are best solved in the time domain, while the shapelet transform domain provides good results for human sensor problems.

Table 7.4: Percentages to show where each transform space produced the lowest error rates, broken down by problem type.

Dataset Type	EE	Shapelets	PS	Change
Human Sensor	20%	80%	0%	0%
Image	46%	21%	18%	14%
Motion	73%	27%	0%	0%
Sensor	35%	39%	17%	9%

We found that problem type alone was not enough to accurately predict the correct transform space, so created a meta-classification problem over the 72 datasets to predict best transform by embedding problem type into a set of other training features, such as component classifier ranks, accuracies, and dataset characteristics. Unfortunately this did not fair much better than using training cross-validation accuracies alone; the meta-classification approach lead to predicting the best transform correctly only 60% of the time. This again lead to a classifier that was significantly less accurate than flat-COTE. It appears as if there is simply too much noise in the training accuracies to accurately estimate the best transform.

7.6.2 Weighted Internal Ensembles

The results of the theoretical oracle-COTE against the flat-COTE provided an interesting finding; though not significant, over the 72 datasets, using an optimistically-biased

classifier that always selected the best transform space did not improve upon the rank of simply using the flat-COTE. This suggests that the enhanced classification performance of the flat-COTE over the EE that was demonstrated in Section 7.3 is not though only finding similarity in a more appropriate data space, but it is actually achieved by finding similarity in multiple domains simultaneously. Clearly the approach of selecting the single-best transform ensemble did not improve the flat-COTE, so the next logical step is to include all ensembles, but employ a hierarchical approach to weight the votes of the ensembles to favour those that indicate they will perform well from the training cross-validation accuracies.

However, a significant drawback of this approach is that it requires a further level of cross validation. For the shapelet transform and EE in particular, this introduces a potentially unacceptable time overhead. For the shapelet transform, we would have to perform shapelet discovery independently on each fold. For the EE, we would have to estimate the parameters for each distance measure independently on each fold. This approach is not a practical solution, so we could alternatively weight according to various summary statistics of the constituent classifiers within each ensemble. We investigated weighting according to mean, median, and mean weighted by variance, but all attempts at hierarchical weighting resulted in a COTE classifier that was significantly worse than the flat-COTE over the 72 problems. This is summarised in the critical difference diagram in Figure 7.7.

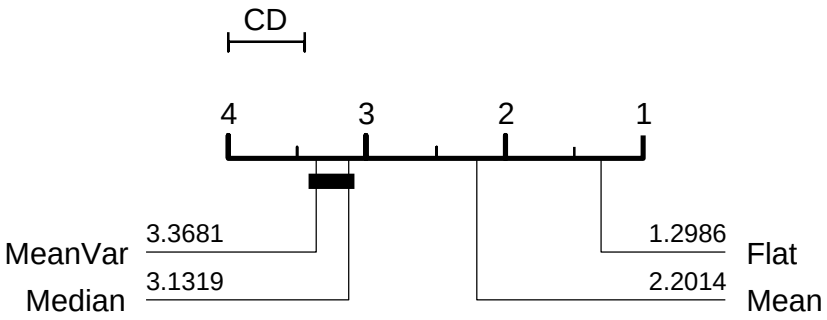


Figure 7.7: Flat-COTE against alternative hierarchical weighted collectives. The flat scheme is significantly more accurate than weighting the ensemble according to any of the simple summary statistics that were tried.

7.6.3 Subset of Internal Ensembles

In pursuit of a more informative configuration for the COTE than the flat structure, thus far we have found that selecting a single transform discards useful information, while weighting is difficult due to the nature of finding unbiased estimate of transform accuracy for the EE and shapelet transform. Regardless, there are clearly classification problems that certain transforms will be inappropriate for, likely reducing classification accuracy by injecting noise into the overall decision process. For example, the case study in Section 7.4 demonstrated that the change transform was the best ensemble for the Caenorhabditis elegans problems, while the EE and shapelet ensemble were relatively poor on this problem. In this example, the change ensemble outperformed flat-COTE, so it would be desirable to omit the EE and shapelet ensembles in this case as it would likely lead to improved test accuracy. Therefore, it is desirable to develop a configuration of the COTE that includes a mechanism for selecting a subset of transforms to use for a dataset based on within-transform classifier variation, with the aim to avoid any disruption caused by the presence of unsuitable classifiers.

We attempt to implement this approach for the COTE by posing the question of whether or not to include a transform as a hypothesis test (in a similar vein to the Significant weighting scheme for the EE that we explored in Section 5.5.2). We initially identify the best transform on the training data, and then evaluate the remaining constituent ensembles to estimate their accuracy on a given dataset. If there is compelling evidence that a constituent will perform significantly worse than the best ensemble, it is pruned from the collective. We implement this scheme by using a two sample Mann-Whitney rank sum test (at the 1% level), where the samples consist of the training accuracies of the constituent classifiers for each ensemble. As a consequence, the resulting ensemble (which we call Mann-COTE) can possibly include either 1, 2, 3, or 4 transforms in the final classifier. Over the 72 datasets, there is no significant difference between Mann-COTE and flat-COTE, with Mann-COTE winning on 25 datasets, flat-COTE best on 23, and they tie on 24 (in half of these cases, the tie is due to no ensemble being pruned from Mann-COTE). To further scrutinise the procedure of the Mann-COTE, Table 7.5 lists these results, cross-referenced by the number of transforms selected by Mann-COTE. The results demonstrates that there is no apparent bias in the number of transforms selected by Mann-COTE, though most often two transforms are selected from the set of four. The results of the Mann-COTE provide an alternative configuration using the collective that retains the accuracy of flat-COTE, while adding greater exploratory power to the results and adds the mechanism for removing unsuitable

data representations. While the latter does not significantly improve the COTE at this stage, we would expect it to be very useful for further work by allowing additional constituent classifiers to be added (or perhaps even further transformation-based ensembles) without fear of overwhelming the decision process.

Table 7.5: Frequency of test set wins for Mann-COTE vs. flat-COTE, according to the number of transforms selected for a given dataset by Mann-COTE.

	Flat Wins	Tie	Mann Wins	Total
One transform	9	3	7	19
Two transforms	12	4	13	29
Three transforms	2	5	5	12
Four transforms	0	12	0	12
Total	23	24	25	72

7.7 Conclusion

In this chapter, we have proposed an ensemble scheme for TSC that is based on constructing classifiers with different data representations, and have shown that this approach is significantly better than all competing algorithms that we have found. The standard baseline algorithms used in TSC research are 1-NN with Euclidean distance and/or Dynamic Time Warping, and we have conclusively shown that COTE significantly outperforms both. We believe that these results represent a new state-of-the-art that new TSC algorithms should be compared against in terms of accuracy, though acknowledge that accuracy is not the only criteria for assessing a classification algorithm. All implementations of the COTE, including flat-COTE, involve many time consuming transformations and cross-validation experiments for parameter setting. However, we also note that the flat-COTE and Mann-COTE are by no means the final evolution of research into forming collectives of transformation-based ensembles. Initially, we could improve the approach through investigating the following:

1. Alternative transforms could be assimilated into the collective if they are found to add diversity. This includes those based on, but not limited to, frequency counts, interval statistics, and complexity measures.

2. We could improve the existing transforms. For example, fields such as speech processing typically use a spectral window rather than transforming whole series. This

could be investigated to offer the possibility of detecting localised discriminatory frequency features for problems with very long series.

3. Our choice of classifiers in the heterogeneous ensemble is fairly arbitrary; only the EE has had a specific emphasis placed on the constituent classifiers. The inclusion of more complex classifiers, the exclusion of weaker classifiers, and the setting of parameters through cross-validation for the heterogeneous ensemble would likely significantly improve the COTE further.

Chapter 8

Conclusions and Future Work

The work in this thesis initially investigated the current benchmark approaches in the field of time series classification (TSC). Using the insight gained from this investigation, a number of novel algorithms were proposed to contribute to the TSC literature, culminating in the proposal of a final classifier that we believe is the first to outperform Dynamic Time Warping on the widely-used UCR time series repository datasets [65].

The aim of this research was to answer the question outlined in Chapter 1: with no prior specialised knowledge about a time series classification task, what is the best way to approach the problem? The majority of research into TSC has considered similarity between series in the time domain, typically coupling elastic distance measures with nearest neighbour (NN) classifiers [58, 79, 8]. However, no such approach has been demonstrated to significantly outperform a 1-NN classifiers with Dynamic Time Warping with a warping window set through cross-validation (DTWCV). This has lead to DTWCV being considered the benchmark for TSC that new algorithms are judged against, with many reiterating that DTWCV with a 1-NN classifier is difficult to beat [9, 21, 47, 34].

Our thesis, built upon the foundations of the work in [3], is that the best way to approach a TSC problem with no prior specialised knowledge is to first transform data into alternative representations where discriminatory features may be more easily detected, and then use these representations to train classification models in multiple domains. We consider the time, frequency, change, and shapelet domains. We believe that the best procedure for building classifiers in these domains is to create an ensemble classifier with each representation, and use these ensembles as constituents in collective of transformation-based ensembles.

8.1 Discussion of Contributions

For classification in the time domain, we proposed a new ensemble classifier: the elastic ensemble (EE). In Chapter 5, we evaluated the common belief that DTWCV is the benchmark in TSC, empirically demonstrating that it is not outperformed by standard classification approaches such as random forest and support vector machines, nor by recently-proposed elastic similarity measures when coupled with 1-NN classifiers. However, while there was no significant difference between the results of the elastic measures, we detected that there was a significant difference in the decisions that they produced. Using this information, we created the EE by combining alternative elastic measures with 1-NN classifiers and tested various weighting schemes, where a proportional weighting scheme produced the most promising results. We then demonstrated that the EE is, to the best of our knowledge, the first time domain classifier to significantly outperform DTWCV over the UCR datasets.

For classification in the shapelet domain, we introduced a novel transformation method using time series shapelets. The original research with shapelets demonstrated good potential with intuitive results, but the implementation limited shapelets to a decision tree approach. We described an algorithm to separate shapelet discovery from the classification algorithm, subsequently introducing the shapelet transform. Firstly, we demonstrated that transforming TSC problems with the shapelet transform and training a decision tree classifier did not produce significantly different results to using the original shapelet tree approach. Secondly, we showed that using shapelet-transformed data with more complex, non-tree-based classifiers lead to significantly better test accuracy than using the shapelet tree. Thirdly, through the use of a case study we demonstrated that the shapelet transform retains the desirable property of intuitive results provided by the original tree-based implementation. Finally, after comparing the shapelet transform to two updated versions of the shapelet tree on an extended set of problems, we demonstrated that training classifiers with shapelet-transformed data still provided significantly better test classification, and provided competitive performance when compared to the EE.

Our final contribution is to use these two lines of study to form a final classifier, COTE: the collective of transformation-based ensembles. COTE is formed by building ensemble classifiers in the four domains that we have considered. In the time domain, we use the EE. In the shapelet domain, we transform data using the shapelet transform and use the resulting data to train a heterogeneous ensemble formed of standard classification approaches. In the change and frequency domains, we use autocorrelation and

spectral transforms respectively to transform data, and also build heterogeneous ensembles in each domain. Using the insight gained when developing the EE, we implement the COTE initially using a flat structure and a proportional weighting scheme. The results achieved from the flat-COTE are very promising; not only does the flat-COTE significantly outperform DTWCV on the UCR data, but it also significantly outperforms the EE too. This result provides strong evidence to support our thesis; by forming the EE in the time domain we were able to significantly outperform the benchmark of DTWCV, and by extending this approach to transform data into alternative representations where discriminatory features may be more easily detected before then building COTE, we were able to significantly improve upon the EE further. Finally, to provide more intuitive results with COTE, we proposed a version of the collective that judges inclusion of constituents on tests of significance. We demonstrated that this version, Mann-COTE, did not provide significantly different results to flat-COTE, while offering more insight into the construction of the classifier.

In conclusion, through extensive experimentation and introduction of new algorithms in the TSC literature, we have produced strong evidence to support our original thesis. Through transforming TSC into alternative domains and building a collective of ensemble classifiers, we have proposed a classifier that is significantly more accurate than any other TSC algorithm that we can find in the literature.

8.2 Future Work and Extensions

The COTE classifier provided very positive test classification results. However, we believe that this performance can be increased further still. One of the key design principles that we have built into COTE is the transparency of the approach. For the four domains that we build classifiers on, each transformation-based ensemble is a modular component where we have ensured that data transformations remain separate from classification models, and all ensembling schemes are transparent. Though COTE is an effective TSC algorithm, these design decisions mean that it is also an easily-extendible framework. For example, due to the modular nature of the transformation-based ensembles within COTE, it is simple to include additional ensembles built in other domains. Also, classifiers within the constituent ensembles can easily be added, exchanged, or removed. Therefore, if we find evidence that a classification algorithm could improve the collective, we can potentially assimilate *any* TSC algorithm into COTE. This could be both at the ensemble level, or as a constituent classifier within an ensemble. In this regard, the potential options for extending COTE are virtually limitless.

In terms of the COTE algorithm that we have proposed in this work however, there are at least two key areas where further research could improve the classifier: time complexity and classifier optimisation. Throughout our work, we have maintained that the best way to evaluate new TSC algorithms is by comparing test accuracy to other approaches. We stand by this statement, but acknowledge accuracy is not the only criteria we could use to assess a new classifier. The runtime of an algorithm may be critical under certain conditions, such as if real-time decisions are required. When proposing COTE, we have not made a significant effort to ensure that the implementation is optimally efficient. For example, in the time domain, we could explore using lower-dimensionality representations of raw data for the EE to observe whether we can achieve the same level of accuracy with less processing. In the shapelet domain, we could investigate an early abandon with quality measure calculations, and research techniques for reducing the number of shapelets that we use for transforming data. For example, if we clustered very similar shapelets together and only retained the best of the matched shapelets, transformed data would have a lower dimensionality without losing discriminatory power. In the change domain, we could investigate whether it is necessary to use ACF, PACF and AR terms, or if using a single autocorrelation-based approach would be sufficient. This would potentially reduce transformed data by up to 66% if we only used one of the three approaches.

A final area where we could focus further research effort is the heterogeneous ensemble that we use in the shapelet, change, and spectral domains. We created the ensemble with a number of leading classification approaches to ensure that it contained strong, diverse constituents. However, the selection process was not as refined as when proposing the elastic ensemble. Therefore we could extend the heterogeneous ensemble to add further algorithms, or investigate whether all constituents contribute to the ensemble and remove any classifiers that potentially have a detrimental effect on performance or runtime. This would not be a simple investigation however, as the heterogeneous ensemble is used in three domains and constituent classifiers may be effective in a subset of the domains. Therefore it may be beneficial to define a separate heterogeneous ensemble for each domain. Additionally, in the EE, we perform cross-validation on the training data to parameterise the elastic distance measures. In the heterogeneous ensemble, we currently use the default Weka [48] parameters when building the classifiers. It is possible that setting parameters such as the number of trees in a random forest, or number of iterations in the rotation forest,could lead to significantly better performance in the heterogeneous ensemble, and COTE as a whole.

Bibliography

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. *Lecture Notes in Computer Science*, 730:69–84, 1993.

[2] J. Aßfalg, H.P. Kriegel, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. Similarity search on time series based on threshold queries. In *Advances in Database Technology-EDBT 2006*, pages 276–294. Springer, 2006.

[3] A. Bagnall, L. M. Davis, J. Hills, and J. Lines. Transformation based ensembles for time series classification. In *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*, pages 307–318. SIAM, 2012.

[4] A. Bagnall and G. Janacek. Clustering time series from arma models with clipped data. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 49–58. ACM, 2004.

[5] A. Bagnall and G. Janacek. A run length transformation for discriminating between auto regressive time series. *Journal of Classification*, 31(2):154–178, 2014.

[6] A. Bagnall and J. Lines. An experimental evaluation of nearest neighbour time series classification. *arXiv preprint arXiv:1406.4757*, 2014.

[7] A. Bagnall, I. Whittley, G. Janacek, K. Kemsley, M. Studley, and L. Bull. A comparison of DWT/PAA and DFT for time series classification. In *Proceedings of the 2006 International Conference on Data Mining (DMIN)*, pages 403–409, 2006.

[8] G. Batista, E. Keogh, O. Tataw, and V. de Souza. CID: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28(3):634–669, 2014.

[9] G. Batista, X. Wang, and E. Keogh. A complexity-invariant distance measure for time series. In *Proceedings of the 11th SIAM International Conference on Data Mining (SDM)*, pages 699–710. SIAM, 2011.

[10] G. Batista, X. Wang, and E. Keogh. A complexity-invariant distance measure for time series. In *Proceedings of the 11th SIAM International Conference on Data Mining (SDM)*, pages 699–710. SIAM, 2011.

[11] M. Baydogan, G. Runger, and E. Tuv. A bag-of-features framework to classify time series. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(11):2796–2802, 2013.

[12] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, pages 359–370, 1994.

[13] M. Bober. Mpeg-7 visual shape descriptors. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):716–719, 2001.

[14] G. Box, G. Jenkins, and G. Reinsel. *Time series analysis: forecasting and control*. John Wiley & Sons, 2013.

[15] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[16] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[17] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. CRC press, 1984.

[18] A. Brown, E. Yemini, L. Grundy, T. Jucikas, and W. Schafer. A dictionary of behavioral motifs reveals clusters of genes affecting *caenorhabditis elegans* locomotion. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 10(2):791–796, 2013.

[19] G. Brown and A. Mood. On median tests for linear hypotheses. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. The Regents of the University of California, 1951.

[20] K. Buza. *Fusion Methods for Time-Series Classification*. PhD thesis, University of Hildesheim, Germany, 2011.

[21] K. Buza, A. Nanopoulos, and L. Schmidt-Thieme. Fusion of similarity measures for time series classification. In *Hybrid Artificial Intelligent Systems*, pages 253–261. Springer, 2011.

[22] J Caiado, N. Crato, and D. Peña. A periodogram-based metric for time series classification. *Computational Statistics & Data Analysis*, 50(10):2668–2684, 2006.

[23] K.P. Chan and A.W.C Fu. Efficient time series matching by wavelets. In *Proceedings of the 15th International Conference on Data Engineering*, pages 126–133. IEEE, 1999.

[24] N. Chawla. C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *Proceedings of the ICML 03 Workshop on Class Imbalances*, volume 3, 2003.

[25] L. Chen and R. Ng. On the marriage of Lp-norms and edit distance. In *Proceedings of the 30th International Conference on Very Large Databases*, pages 792–803. VLDB, 2004.

[26] L. Chen, M. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 491–502. ACM, 2005.

[27] M. Corduas and D. Piccolo. Time series clustering and classification by the autoregressive metric. *Computational Statistics & Data Analysis*, 52(4):1860–1872, 2008.

[28] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[29] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *Principles of Data Mining and Knowledge Discovery*, pages 88–100. Springer, 1997.

[30] L. M. Davis. *Predictive Modelling of Bone Ageing*. PhD thesis, University of East Anglia, United Kingdom, 2013.

[31] L. M. Davis, B.J. Theobald, J. Lines, A. Toms, and A. Bagnall. On the segmentation and classification of hand radiographs. *International Journal of Neural Systems*, 22(05), 2012.

[32] L. M. Davis, B.J. Theobald, A. Toms, and A. Bagnall. On the extraction and classification of hand outlines. In *Intelligent Data Engineering and Automated Learning-IDEAL 2011*, pages 92–99. Springer, 2011.

[33] J Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[34] H. Deng, G. Runger, E. Tuv, and M. Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239, 2013.

[35] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. In *Proceedings of the 34th International Conference on Very Large Databases*. VLDB, 2008.

[36] J. Durbin. The fitting of time-series models. *Revue de l'Institut International de Statistique*, pages 233–244, 1960.

[37] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*, volume 57. CRC press, 1994.

[38] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.

[39] E. Fix and J. L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: small sample performance. Technical report, DTIC Document, 1952.

[40] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[41] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[42] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th Internaltional Conference on Machine Learning (ICML)*, volume 96, pages 148–156, 1996.

[43] B. Fulcher and N. Jones. Highly comparative feature-based time-series classification. *arXiv preprint arXiv:1401.3531*, 2014.

[44] E. Gabrilovich and S. Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5. In *Proceedings of the 21st Internaltional Conference on Machine Learning (ICML)*.

[45] P. Geurts. Pattern extraction for time series classification. In *Principles of Data Mining and Knowledge Discovery*, pages 115–127. Springer, 2001.

[46] T. Górecki and M. Łuczak. Using derivatives in time series classification. *Data Mining and Knowledge Discovery*, 26(2):310–331, 2013.

[47] J. Grabocka, A. Nanopoulos, and L. Schmidt-Thieme. Invariant time-series classification. In *Machine Learning and Knowledge Discovery in Databases*, pages 725–740. Springer, 2012.

[48] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[49] B. Hartmann and N. Link. Gesture recognition with inertial sensors and optimized DTW prototypes. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 2102–2109. IEEE, 2010.

[50] Q. He, Z. Dong, F. Zhuang, T. Shang, and Z. Shi. Fast time series classification based on infrequent shapelets. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 1, pages 215–219. IEEE, 2012.

[51] J. Hills. *Mining Time-series Data using Discriminative Subsequences*. PhD thesis, University of East Anglia, Norwich, 2015.

[52] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.

[53] D. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, 24(4):664–675, 1977.

[54] C. A. R. Hoare. Algorithm 65: find. *Communications of the ACM*, 4(7):321–322, 1961.

[55] B. Hu, Y. Chen, and E. Keogh. Time series classification under more realistic assumptions. *Proceedings of the 13th SIAM International Conference on Data Mining (SDM)*, pages 578–586, 2013.

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

[57] G. Janacek, A. Bagnall, and M. Powell. A likelihood ratio distance measure for the similarity between the Fourier transform of time series. In *Advances in Knowledge Discovery and Data Mining*, pages 737–743. Springer, 2005.

[58] Y. Jeong, M. Jeong, and O. Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44:2231–2240, 2011.

[59] T. Kahveci and A. Singh. Variable length queries for time series data. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 273–282. IEEE, 2001.

[60] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[61] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[62] E. Keogh and M. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 285–289. ACM, 2000.

[63] E. Keogh and M. Pazzani. Derivative dynamic time warping. In *Proceedings of the 1st SIAM International Conference on Data Mining (SDM)*, pages 5–7. SIAM, 2001.

[64] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

[65] E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. Ratanamahatana. The UCR time series classification/clustering homepage. `http://www.cs.ucr.edu/ eamonn/time_series_data/`, 2011.

[66] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, volume 14, pages 1137–1145, 1995.

[67] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *ACM SIGMOD Record*, 26(2):289–300, 1997.

[68] W. Kruskal and W. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[69] L.J. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 424–429. IEEE, 2000.

[70] N. Levinson. The wiener RMS error criterion in filter design and prediction, appendix b of wiener, n.(1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*, 1949.

[71] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research Ossues in Data Mining and Knowledge Discovery*, pages 2–11. ACM, 2003.

[72] J. Lin, R. Khade, and Y. Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315, 2012.

[73] J. Lines. Supporting resources and results for this thesis. `https://sites.google.com/site/jasonlinesphdthesis/`, 2014.

[74] J. Lines and A. Bagnall. Alternative quality measures for time series shapelets. In *Intelligent Data Engineering and Automated Learning-IDEAL 2012*, pages 475–483. Springer Berlin Heidelberg, 2012.

[75] J. Lines and A. Bagnall. Ensembles of elastic distance measures for time series classification. In *Proceedings of the 14th SIAM International Conference on Data Mining (SDM)*, pages 524–532. SIAM, 2014.

[76] J. Lines and A. Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, pages 1–28, 2014.

[77] J. Lines, A. Bagnall, P. Caiger-Smith, and S. Anderson. Classification of household devices by electricity usage profiles. In *Intelligent Data Engineering and Automated Learning-IDEAL 2011*, pages 403–412. Springer Berlin Heidelberg, 2011.

[78] J. Lines, L. M. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 289–297. ACM, 2012.

[79] P.F. Marteau. Time warp edit distance with stiffness adjustment for time series matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):306–318, 2009.

[80] A. Mueen, E. Keogh, and N. Young. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, pages 1154–1162. ACM, 2011.

[81] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and M. B. Westover. Exact discovery of time series motifs. In *Proceedings of the 9th SIAM International Conference on Data Mining (SDM)*, pages 473–484. SIAM, 2009.

[82] S. Mukherjee, E. Osuna, and F. Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 511–520. IEEE, 1997.

[83] K.R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In *Artificial Neural NetworksICANN'97*, pages 999–1004. Springer, 1997.

[84] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276–285. IEEE, 1997.

[85] Image Processing and University of Southern California Informatics Lab. The digital hand atlas database system. `http://www.ipilab.org/BAAweb//`, 2007.

[86] J. R. Quinlan. *C4.5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[87] J. R. Quinlan. Improved use of continuous attributes in C4.5. *arXiv preprint cs/9603103*, 1996.

[88] T. R. and E. Keogh. Fast-shapelets: A fast algorithm for discovering robust time series shapelets. In *Proceedings of the 13th SIAM International Conference on Data Mining (SDM)*. SIAM, 2013.

[89] T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proceedings of the 13th SIAM International Conference on Data Mining (SDM)*. SIAM, 2013.

[90] C. Ratanamahatana and E. Keogh. Three myths about dynamic time warping data mining. In *Proceedings of the 5th SIAM International Conference on Data Mining (SDM)*, pages 506–510. SIAM, 2005.

[91] C. A. Ratanamahatana and E. Keogh. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data*, pages 22–25, 2004.

[92] C. A. Ratanamahatana and E. Keogh. Making time-series classification more accurate using learned constraints. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM)*, pages 11–22. SIAM, 2004.

[93] J. Rodriguez and C. Alonso. Support vector machines of interval-based features for time series classification. *Knowledge-Based Systems*, 18:171–178, 2005.

[94] J.J. Rodriguez, L.I. Kuncheva, and C.J. Alonso. Rotation forest: A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1619–1630, 2006.

[95] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.

[96] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[97] K.M. Schneider. A comparison of event models for Naive Bayes anti-spam e-mail filtering. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 307–314. Association for Computational Linguistics, 2003.

[98] C. E. Shannon. A mathematical theory of communication. *The Bell System Technichal Journal*, 27:379–423, 623–656, 1948.

[99] D. Silva, V. de Souza, and G. Batista. Time series classification using compression distance of recurrence plots. In *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM)*, pages 687–696. IEEE, 2013.

[100] J. Slapin and S.O. Proksch. A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, 52(3):705–722, 2008.

[101] A. Stefan, V. Athitsos, and G. Das. The move-split-merge metric for time series. 25(6):1425–1438, 2012.

[102] J. Tanner, R. Whitehouse, M. Healy, H. Goldstein, and N. Cameron. Assessment of skeletal maturity and prediction of adult height (TW3) method. In *Academic Press*, 2001.

[103] F. Tay and L. Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317, 2001.

[104] Energy Saving Trust. *Powering the Nation.* Department for Environment, Food and Rural Affairs (DEFRA), 2012.

[105] D. Wolpert and W. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[106] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 1033–1040. ACM, 2006.

[107] L. Ye and E. Keogh. Time series shapelets: A new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956, 2009.

[108] L. Ye and E. Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 22(1-2):149–182, 2011.

[109] E. Yemini, T. Jucikas, L. Grundy, A. Brown, and W. Schafer. A database of *caenorhabditis elegans* behavioral phenotypes. *Nature Methods*, 10:877–879, 2013.

[110] B.K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *Proceedings of the 26th International Conference on Very Large Databases*. VLDB, 2000.