Shapelet Transforms for Univariate and Multivariate Time Series Classification

Aaron George Bostrom

A thesis submitted for the
degree of Doctor of Philosophy

University of East Anglia
School of Computing Sciences

May 2018

Abstract

Time Series Classification(TSC) is a growing field of machine learning research. One particular algorithm from the TSC literature is the Shapelet Transform (ST). Shapelets are phase independent subsequences that are extracted from time series to form discriminatory features. It has been shown that using the shapelets to transform the dataset into a new space can improve performance. One of the major problems with ST, is that the algorithm is $O(n^2m^4)$, where n is the number of time series and m is the length of the series. As a problem increases in size, or additional dimensions are added, the algorithm quickly becomes computationally infeasible.

The research question addressed is whether the shapelet transform be improved in terms of accuracy and speed. Making algorithmic improvements to shapelets will enable the development of multivariate shapelet algorithms that can attempt to solve much larger problems in realistic time frames.

In support of this thesis a new distance early abandon method is proposed. A class balancing algorithm is implemented, which uses a one vs. all multi class information gain that enables heuristics which were developed for two class problems. To support these improvements a large scale analysis of the best shapelet algorithms is conducted as part of a larger experimental evaluation. ST is proven to be one of the most accurate algorithms in TSC on the UCR-UEA datasets. Contract classification is proposed for shapelets, where a fixed runtime is set, and the number of shapelets is bounded. Four search algorithms are evaluated with fixed run times of one hour and one day, three of which are not significantly worse than a full enumeration. Finally, three multivariate shapelet algorithms are developed and compared to benchmark results and multivariate dynamic time warping.

Acknowledgements

First and foremost I would like to thank my supervisor, Dr. Anthony Bagnall, whose continued patience and support has proved invaluable throughout this process. I would like to thank my friends and colleagues at UEA and especially those in the time series classification group who have supported me.

I would also like to extend my thanks to Dr. Ji Zhou, and his team over at the Earlham Institute, he has supported my development as a researcher during my write up period, and I look forward to working with the group as I begin the next stage of my career.

Most importantly I would like to thank my beautiful fiancé Amy Fellows, who has been the mental and physical support I have needed during my PhD, patiently putting up with the long nights, and strange working hours, and of course my cat Nacho for staying up with me.

Finally I would like to thank my good friends, to my teacher Paul Fretter, my friends George Beard, Adam Garner, Hilton Pashley, Sophie Farenden, James Large, James Macnamara, Leo Wilkins, Joshua Ball, Danny Reynolds and Pratik Gurung you have been a source of laughter and fun and have helped me immensely and I cannot thank you all enough.

List of Figures

List of Tables

List of Algorithms

Contents

List of Publications

As First Author

- A. Bostrom and A. Bagnall. Binary shapelet transform for multiclass time series classification. *Proc. 17th International Conference on Big Data Analytics and Knowledge Discovery (DAWAK)*, 2015
- A. Bostrom, A. Bagnall, and J. Lines. Evaluating improvements to the shapelet transform. *Knowledge Discovery and Data Mining, in Workshop on Mining and Learning from Time Series*, 2016
- A. Bostrom and A. Bagnall. Binary shapelet transform for multiclass time series classification. *Transactions on Large-Scale Data and Knowledge Centered Systems XXXII: Special Issue on Big Data Analytics and Knowledge Discovery*, pages 24–46, 2017
- A. Bostrom and A. Bagnall. A Shapelet Transform for Multivariate Time Series Classification. *ArXiv e-prints*, 2017

As Co-author

- A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with COTE: The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27:2522–2535, 2015
- A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advance. *Data Mining and Knowledge Discovery*, pages 1–55, 2016

Chapter 1

Introduction

1.1 Introduction

Shapelets are subsequences of time series that are phase independent discriminatory features for Time Series Classification (TSC). TSC is a growing area of machine learning research. We consider time series data to be any ordered real-valued data. Time series classification is a specialization of the general classification problem. We define classification as: given a set of inputs $\mathbf{x}$ and outputs y, can we find a mapping from $\mathbf{x}$ to y? We define y as a set of unique labels where $y \in \{1, ..., C\}$. We assume that $y = f(\mathbf{x})$ for some unknown function f, and the goal of classification is to learn f from a set of labelled inputs so that $\hat{y} = \hat{f}(\mathbf{x})$. Informally the goal is to learn from the labelled training data so that we can predict class membership of unknown series.

Classification relies on finding or deriving explanatory features, either through a probabilistic approach, or by measuring similarity to form groupings and define decision boundaries. In traditional classification, and in simpler models, such as naïve bayes, attributes are often treated as independent of one another. However, in TSC problems the ordering of the attributes can be critical in deriving explanatory features, and in being able to discriminate between classes. There has been a large amount of research

into algorithms for time series classification, some of which we review in chapter 2. Recently a large scale experimental evaluation [8] compared the best algorithms from the literature and found that transformation based approaches performed better on average. One of the best algorithms in this study was the shapelet transform [70].

1.2 Motivation

The shapelet transform (ST) was proposed in [70] where it was adapted from the the shapelet tree algorithm [114]. Shapelets are phase independent subsequences that are found within the time series data and they are covered in great detail in chapter 2. ST was shown to be a significant improvement over the tree based implementation, as the model was uncoupled from a decision tree and a better classification algorithm was paired with this shapelet transformed data.

One of the major problems with the shapelet tree algorithm, and subsequently the shapelet transform algorithm, was the run time. Shapelets were originally created because they capture phase independent features. These features could also be mapped back to the original series to derive data driven rules that could be interpreted by a human. The problem with this approach is that to find the global best features, a full enumeration of all subsequences in the datasets is required, which is very time consuming. There are three major motivations for this thesis. Firstly, to improve the classification accuracy of the shapelet transform by extracting better quality shapelets. In [72] different quality measures were evaluated for extracting shapelets. These methods can be problematic on multi-class problems when the number of classes is very high. The distribution of shapelets that can be found is largely dependent on the underlying class distribution of the training data. Where there is an imbalance in the distribution of classes in a dataset, this can adversely affect the number of shapelets found, as such we may need to compensate for this an evenly distribute the number of shapelets on a per class basis.

The second motivation for this thesis is to drastically reduce the runtime

of the shapelet transform. The current runtime complexity of the shapelet transform is $O(n^2m^4)$ where n is the number of time series and m is their length. In chapter 2 we cover the early abandon techniques already proposed in the literature, the objective is to improve upon existing techniques. In other areas of Computer Science research time constrained algorithms exist for approximating difficult to solve problems. Shapelet finding could be an ideal candidate for heuristic search algorithms and time constrained learning.

The third motivation is to adapt ST for multivariate time series classification. Multivariate time series data is becoming widespread, where the number of sensors and devices are able to capture vast quantities of data. One particular area of multivariate time series classification research is electroencephalography (EEG). Critically, very little shapelet based research has been conducted on multivariate data. Shapelets as a technique are only defined in the univariate case, the motivation is to define shapelets for the multivariate case, and leverage off the previous improvements to efficiency to enable multivariate TSC. The time complexity problem is only exacerbated as more data points are added. *No free lunch theorem* is present in many fields, and time series classification is no exception [111]. No single algorithm will generalise well to all problems and tailor made algorithms for multivariate time series classification are required. However, we are motivated to assess current simpler approaches to see where they can compete with more hand-crafted solutions and provide benchmark results with which to compare.

1.3 Contributions

To provide support for this thesis, large scale experimentation were conducted and novel algorithms are proposed. The contributions of this thesis are as follows:

- **Binary shapelet transform for multi-class time series classification**. We present our novel algorithm for balancing binary shapelets, which leverages existing speed up techniques on multi-class problems.

A revised evaluation order is shown to reduce the number of fundamental operations required in the average case compared to existing speed up techniques. A study of the multi-class datasets found in the UCR-UEA archive [23] found that the balanced shapelets improve the shapelet transform on multi-class problems. We present the concept of a shapelet transform that uses balancing and binary shapelets when the number of classes is greater than two, or otherwise reverts to the original. This work is reported in chapter 4 and published in [15, 16].

- **The great time series classification bake off**. This was a large experimental evaluation undertaken by the research group at UEA. An endeavour to implement the 20 most common algorithms from the TSC literature under a common framework and evaluate them on 8500 datasets. The contribution presented in this thesis in chapter 4 and used extensively in further comparisons in chapter 5 was implementing and testing the Learn Shapelets and Fast Shapelets algorithm on 8500 problems [83, 40] and comparing them with the Shapelet Transform presented in the first portion of chapter 4. These results were published in [8] where the Shapelet Transform was only beaten by COTE, of which the shapelet transform is a constituent. The Shapelet results contributed to the building of the COTE ensemble and the changes made to ST contributed in part to the improvements seen from the previous iteration which was presented in [6].

- **Evaluating the shapelet transform**. A converted Fast Shapelet algorithm is presented as a transform instead of a decision tree, and shown that it is significantly better than the tree implementation. However, the Fast Shapelet Transform is significantly worse than the Shapelet Transform, although it is considerably faster. We then present a contract shapelet algorithm where the stride parameters can be derived from a given time limit. These stride parameters enable the shapelet search to avoid areas of the search space and constrain it to a fixed runtime. It is shown that heuristically evaluating the search space is not significantly worse than a full enumeration and the work published

in [18] is improved upon by considering more complex heuristic search techniques in chapter 5.

- **Shapelet Transform for Multivariate Time Series Classification**. Three novel approaches to multivariate time series classification are described. These multidimensional shapelet algorithms are benchmarked against a number of common machine learning algorithms on multivariate datasets. Univariate classification algorithms are adapted to the multivariate data by either concatenating the dimensions into a single series or by forming a homogeneous ensemble on each dimension. We have sourced and processed 24 datasets from the literature and converted them into a common format for use with the WEKA framework, building a foundation for the MTSC community to expand upon. The work and results are publicly available and are published in e-print [17], whilst also being under review.

1.4 Thesis Organisation

The remainder of this thesis is organised as follows. In chapter 2 a review of the time series classification literature, with a large emphasis on shapelet research is presented. In chapter 3 we describe the datasets we use to benchmark with, the way in which we conduct large scale experiments and some of the statistics we use to present and analyse our results. In chapter 4 we outline our first contribution, which aims to reduce the runtime of the shapelet transform and increase the classification accuracy on multi-class problems. In the second portion of chapter 4 we discuss our contribution to the work [8] and how these form the core of experimental methodology for later contributions, as well as the benchmark which we aim to maintain in chapter 5. The aims of chapter 5 are to build on the success of previous work and apply heuristic techniques from both the literature and a novel approach to finding shapelets in a fixed time frame. The final contribution is presented in chapter 6 where the speed up techniques developed in previous work are used in conjunction with the three novel shapelet approaches to

multivariate time series classification. This thesis is concluded in chapter 7 where the contributions are discussed and future work is considered.

Chapter 2

Technical Background and Related Work

This chapter introduces some of the technical background used in this thesis. We introduce the problem of time series classification (TSC) and present a review of shapelet based techniques. The aims of this project are to improve Shapelet based classification as it was identified as one of the best time series classification algorithms from within the literature.

In this chapter we present a review of the history of the algorithm, and the various changes and optimisations published since its creation.

2.1 Time Series Classification

There are many types of problems that exist in time series data mining including clustering, classification, querying, forecasting and indexing. In this thesis the sole aim is to focus on shapelet based techniques for time series classification, where the class label is a constant singular value per series.

We define a set of n time series as

$$\mathbf{T} = \{T_1, T_2, ..., T_n\}$$

where each series consist of m real-valued attributes

$$T_i =< t_1, t_2, ..., t_m >$$

and a class value c_i.

Many TSC algorithms are based on measuring similarity between series. There are three major types of similarity in time series classification. These are similarity in time, similarity in shape, and similarity in change. Similarity in time is predominantly found using nearest-neighbour techniques with either Euclidean Distance (ED), or Dynamic Time Warping (DTW) [75, 51, 101, 85, 59, 60, 21]. Similarity in change is where the features of a dataset are embedded in the autocorrelation structure of the time series. An example is an Autoregressive Moving Average Model (ARMA) [25, 3]. Finally, there is similarity in shape. This is the major focus of this thesis. If the shape is local and embedded in the time series, subsequence techniques are needed [40, 114, 70, 83, 79].

2.2 Time Series Classification Algorithms

Bagnall et al. [8] conduct a thorough analysis of a large portion of the algorithms presented in the literature. The major algorithms are broadly separated into six simple groupings. The techniques are grouped into the following categories:

2.2.1 Whole series

Whole series algorithms tend to be based around adaptations and extensions of either the Euclidean distance (ED) or Dynamic Time Warping (DTW) [86]. These algorithms are paired with a nearest neighbour classifier and have seen relative success in large scale experimental comparisons [8]. Many variations of dynamic time warping exist and are covered thoroughly in [8]. Weighted Dynamic Time Warping (WDTW) was presented in [51] where they add a penalty to the warping distance. Time Warp Edit (TWE) was proposed to give a stiffness parameter to the warping [75]. Move-Split-Merge (MSM) is a

metric that is similar to other edit distance algorithms [101]. Other metrics include Edit distance with Real Penalty (ERP) [22] and Longest Common SubSequence (LCSS) [48]. Wang et al. [107] found that over 38 datasets 8 of these measures were not significantly better than DTW.

2.2.2 Intervals

Interval algorithms are described as finding phase dependent features. One of the main interval based approaches is Time Series Forest (TSF) [30]. The main problem with the phase dependent models is that the feature space is very large, they overcame this by using a random forest type approach. Each tree is generated with $\sqrt{m}$ random intervals. These intervals are used to generated summary statistics which are used to build the tree, and classification is majority voting within the ensemble. Time Series Bag of Features (TSBF) [12] and Learned Pattern Similarity (LPS) [11] were proposed as extensions to TSF by the same group at Arizona University. On the UCR archive these methods were found to be not significantly better than each other.

2.2.3 Shapelets

These algorithms find phase independent subsequeneces from within the time series. Essentially these algorithms select subsets of contiguous features and build standard classification models on the features. Shapelets are the main focus of this thesis and the remaining sections in this chapter are dedicated to a large scale review of the shapelet based literature.

2.2.4 Dictionary based

Dictionary based classifiers are broadly based around the Bag Of Patterns model (BOP) which was proposed by [68]. BOP is a dictionary based classifier built on SAX [67]. SAX is covered in greater detail in section 2.7. The distribution of the SAX words in a series produce a histogram of the counts. The same data transformation is applied to new series, and a nearest neighbour of the histograms is used to classify. Symbolic Aggregate

approXimation-Vector Space Model (SAXVSM) combines SAX and a vector space model that is common in Information Retrieval. SAXVSM forms frequencies over classes rather than series, and uses Term-Frequency Inverse Document Frequency (TFIDF) to weight these histograms [95]. Bag of SFA symbols (BOSS) is different to BOP and SAXVSM in that is uses a Discrete Fourier Transform (DFT) instead of a Piecewise Aggregate Approximation (PAA) on each window [93] . The series are truncated using Multiple Coefficient Binning (MCB) rather than the fixed interval approaches of the previous sections. From the literature on the published 19 UCR datasets used, BOP and SAXVSM were not significantly better than each other, and both are significantly worse than BOSS.

2.2.5 Combinations

These algorithms combine one or more of the above approaches into a single classifier.

2.2.6 Model based

These types of algorithms are not well represented in the literature but they include auto-regressive models [9, 25], hidden Markov models [100] and kernel models [24]. These algorithms tend not to be used in classification [74].

2.3 Shapelets

Shapelets are a subsequence of a time series designed for finding local phase independent similarity. They were first proposed in [114] and have been a prominent area of TSC research since. We define a subseries of a time series of length l as a contiguous set of values from within a series T_i. Any contiguous series in a time series can be a shapelet, and so the maximum number of distinct shapelets in a single series is $(m - l + 1)$.

2.4 Shapelet Tree

The shapelet tree algorithm was one of the first algorithms in TSC aimed at finding similarity in shape [114, 115]. The brute force algorithm for finding shapelet and evaluating shapelets is presented in algorithm 1. The algorithm searches through the entire set of subseries within a dataset, evaluating the quality of each subseries, recording the best. The best subseries is selected as the shapelet which is then used as splitting rule in the decision tree, and the process continues on each sub-tree. The data is subdivided by the shapelet at each node until either a maximum depth is reached, or a sub-tree dataset contains all of one class.

Quality and Distance Measures

Information Gain is used to measure the quality of a single shapelet [97]. To calculate information gain a measure of similarity between shapelets and between a shapelet and a series is required. Euclidean distance is used to measure the similarity of two equal length series. Euclidean distance is defined in Equation 2.1, where A and B are series and they are of length l .

$$dist(A, B) = \sqrt{\sum_{i=1}^{l} (A_i - B_i)^2} \tag{2.1}$$

In order to measure the distance between two series that are different lengths we need to define a separate function. $sDist(S, T)$ is a function that uses a sliding window on the longer series, in this case T, where the width of the sliding window is set to that of the shorter series. Each subseries in T is compared to the input subseries S. As these subsequences are the same length, they are compared using the Euclidean distance. This generates a set of distances W which contains $m - l + 1$ distance values. $sDist$ finds the best matching location in the longer series, and thus the smallest distance is considered the best. To illustrate this point, if S is extracted from T, $sDist(S, T)$ should return 0 as the best matching location should be itself.

$$sDist(S,T) = min_{w \in W}(dist(s,w))$$

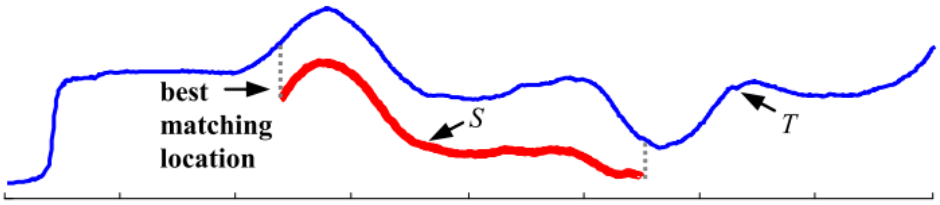


Figure 2.1: sDist diagram taken from Time-Series Shapelets [114].

2.4.1 Information Gain

To evaluate the quality of a single shapelet, information gain [97] is used as a measure of how well it can separate two classes. Initially we present the formula for calculating information gain, and entropy, we then apply this to shapelets with a worked example.

Given a dataset D, with two classes C_1 and C_2, the proportion of time series which belong to class C_1 is $p(C_1)$ and the proportion which belong to C_2 is $p(C_2)$, we define the entropy of D as:

$$E(D) = -p(C_1)log(p(C_1)) - p(C_2)log(p(C_2))$$

To determine the information gain we need to calculate the best split in the dataset D, we create two subsets D_1 and D_2 by splitting D and then calculating the entropy of D_1 and D_2 respectively. The entropy is calculated in proportion to the whole, so calculating the fraction of classes in D_1 as $f(D_1)$ and the fraction of classes in D_2 as $f(D_2)$, the entropy of the split is:

$$\hat{E}(D) = f(D_1)H(D_1) + f(D_2)H(D_2)$$

Given the definition of Entropy and how entropy of a given split is calculated. Information gain can be defined as the difference in the entropy

of the original set compared to the entropy of the two subsets.

$$I = E(D) - \hat{E}(D)$$

2.4.2 Shapelet Quality

The quality of a shapelet is determined by using the distance from the shapelet candidate to every series in the dataset. This generates a list of n distance values, which is called an orderline. An orderline consists of a pair of values, the distance value, and the class label for the respective series, and is sorted in ascending order based on the distance value. An ideal shapelet should produce small distance values when compared to time series of the same class and large distance values with other classes. The optimal configuration for an orderline is where all of the distance and value pairs in the orderline that are the same as shapelets class are located in D_1 and all other pairs are located in D_2.

A given orderline O should contain n distance and class value pairs. The orderline is sorted by the distance into ascending order. The number of splitting points that will generate unique information gain values is $n - 1$. The shapelet algorithm calculates the information gain for all split points, selecting the maximum information gain, and corresponding split point. For clarity, a worked example is described. An orderline of 6 distances with 4 of class B and 2 of class R is illustrated in Figure 2.2. The optimal split point is shown with a dashed line. The orderline splits the 6 distances into two sets. There are 3 total distances on the left and there are 3 total distances on the right. The left hand side contains the elements in D_1 of the equation. This side is simpler as there is only class B present which means there is only 3 of class B. The right hand side contains the elements in D_2 of the equation, there are both classes B and R, where there are 2 of class R, and 1 of class B. In Equation 2.2 the maximum informatio gain of the orderline from Figure 2.2 is calculated. The first section of the equation calculates the portion each class contributes towards the total set. This could also be described as the ideal entropy.

$$
\begin{aligned}
I = [-(4/6)log(4/6) - (2/6)log(2/6)] \\
-(3/6)[-(3/3)log(3/3)] \\
+(3/6)[-(2/3)log(2/3) - (1/3)log(1/3)]
\end{aligned} \tag{2.2}
$$

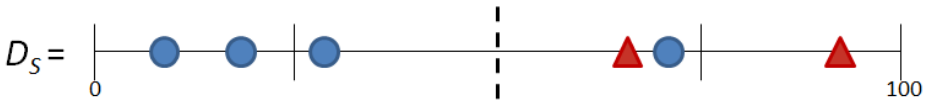


Figure 2.2: Simple orderline with two classes

2.4.3 Brute Force Search

Algorithm 1 FindBestShapelet($\mathbf{T}, min, max$)

Where $\mathbf{T}$ is a set of Time Series.
$best\_quality, quality$
$best\_shapelet, shapelet$
for $\mathbf{T}_i$ in $\mathbf{T}$ **do**
  for $l = min$ **to** max **do**
    for $p = 0$ **to** $|T_i| - l + 1$ **do**
      $shapelet = T_{i,p}^{l}$
      $quality$ =checkCandidate($\mathbf{T}$,$shapelet$)
      if $quality > best\_quality$ **then**
        $best\_quality = quality$
        $best\_shapelet = shapelet$
return $best\_Shapelet$

Algorithm 2 checkCandidate($\mathbf{T}, S$)

Where $\mathbf{T}$ is a set of time series and S is a shapelet candidate.
Where O is an orderline.
for T_i in T **do**
  $dist = sDist(S, T_i)$
  $O \cup < dist, c_i >$
return informationGain(O)

Algorithm 3 sDist(T, S)

Where T is a time series and S is a shapelet candidate.
$l = |S|$
$min\_dist = \infty$
for $p = 0$ **to** $|T| - l + 1$ **do**
  $dist = dist(S, T_p^l)$
  if $dist < min\_dist$ **then**
    $min\_dist = dist$
return min_dist

The brute force search defined in algorithm 1 is slightly modified in contrast to the version in the the original paper [115]. This is to make it more in line with the implementation. The original algorithm description precomputed and stored all the possible shapelet candidates in a set. Instead the algorithm finds and evaluates each shapelet individually.

2.5 Logical Shapelets

Logical shapelets were an adaptation to the shapelet tree algorithm, Mueen et al. [79] proposed a statistics caching optimisation which is discussed in greater detail in subsection 2.11.1. The aim of logical shapelets is to combine shapelets to form more complex rules to better handle difficult to separate problems. The algorithm is a combination of shapelets used in conjunction with each other for determining the class separation on the orderline. In Figure 2.3 we illustrate one of the motivating examples which was presented in the original paper. In the diagram the first class (yellow) has two independent shapes that represent the class, but only where they appear together. The problem with distinguishing this from the other class is that the shapelets occurrence is independent from one another, therefore they cannot be represented as single shapelet and individually they cannot separate either class, thus producing a poor information gain value. The splits that are detected are then divided into either broken or non-linearly separable shapelets. These broken shapelets are evaluated with the logical shapelets, where additional shapelets are searched for to try and achieve a

better splitting.

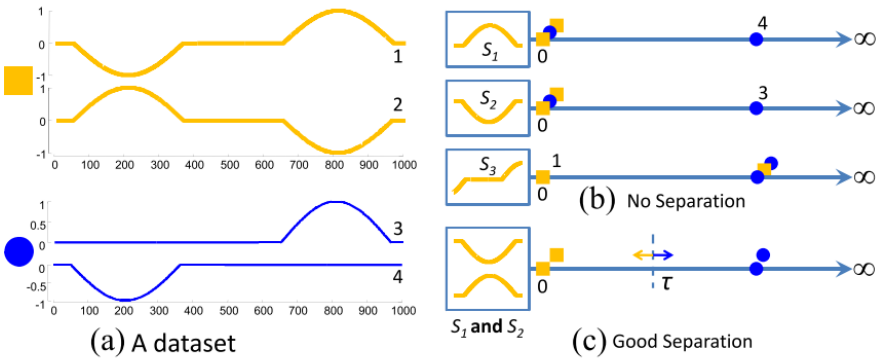


Figure 2.3: Image taken from Logical Shapelets [79].

2.6 Shapelet Transform

The shapelet transform was proposed in [70] and further expanded upon in [47, 72]. A number of significant changes to the original algorithm were proposed, these included: separating the shapelet finding process from the classification model, considering alternative similarity measures and further speed up techniques to the sDist function. As was discussed earlier, the original shapelet algorithm was embedded in a decision tree, finding the best shapelet at each node recursively subdividing the data. This results in the brute force search being performed a number of times at each node, which makes it intractable on large problems. The shapelet transform algorithm does not change the brute force, but requires it is done only once. Separating the shapelet finding algorithm from classification meant that a number of the drawbacks of decision trees could be avoided. Decision trees are often out performed by other classifiers and they have a tendency to over fit unless post-pruned. The shapelet transform performs a data transformation by finding a set of k shapelets and creating a new dataset of k features per series. In algorithm 4 the single pass shapelet search for the k best shapelets

is proposed.

Algorithm 4 FindKBestShapelets($\mathbf{T}$, min, max, k)

Where $\mathbf{T}$ is a set of Time Series.
$KShapelets = \emptyset$
for T_i in $\mathbf{T}$ **do**
  $seriesShapelets = \emptyset$
  for $l = min$ **to** max **do**
    for $p = 0$ **to** $|T_i| - l + 1$ **do**
      $quality = checkCandidate(\mathbf{T}, T^l_{i,p})$
      $seriesShapelets = seriesShapelets \cup \{\mathbf{T}^l_{i,p}, quality\}$
  $sort(seriesShapelets)$
  $removeSelfSimilar(seriesShapelets)$
  $kShapelets = merge(k, kShapelets, seriesShapelets)$

2.6.1 Changes from Shapelet Tree to Shapelet Transform

Algorithm 4 describes the shapelet transform. This section describes the changes from the shapelet tree algorithm in more detail. The brute force search algorithm is the same as the one presented in algorithm 1. The tuning parameter k is the size of the final shapelet set, the other change is the forming of the shapelet set. As we consider each shapelet we create a list, which after each series has been considered, is sorted, and the self similar shapelets are pruned, and merged into the k best shapelets list. This process happens for each series. Self similar shapelets are formally defined in [47]. Informally, self similar shapelets are overlapping subsequences of varying lengths and starting positions. This ensures the best quality and longer non-overlapping shapelets are only considered when merging into the kShapelets set.

2.6.2 Alternative Quality Measures

Lines and Bagnall [72] proposed a number of alternative measures for assessing the quality of shapelets [72]. One of the problems with information gain is that the entropy pruning speed up, presented earlier in subsection 2.11.1, is not efficient on multi-class problems. Calculating the best possible configurations

of a two class problem is possible in constant time because the orderline is 2 dimensional. To find the best possible configuration of multiple classes becomes untenable and does not improve speed. Three alternative distance measures were proposed in [72], these were Kruskal-Wallis, F-statistic(F-stat) and Mood's median [78, 62].

Given a set of n samples F-stat is used to analyse the variance in the difference of means. The statistic is used in Shapelets to test the variability of the distance between Shapelets and series, where low variability of series in the same class, and high variability between classes yields a good shapelet. The set of distances O, our orderline is still required, thus the F-stat is not a complexity improvement, but it has been shown to be more effective than IG. Given our orderline of distances, we sort the distances and their class membership into separate sets, O_i, we also calculate the average distance in O_i as $\bar{O}_i$ and the average distance to all series as $\bar{O}$. We denote the number of classes as C and n is the number of series. We calculate the F-stat value by:

$$F = \frac{\sum_{i} (\bar{O}_i - \bar{O})^2/(C-1)}{\sum_{i=1}^{C} \sum_{d_j \in O_i} (d_j - \bar{O}_i)^2/(n-C)}$$

Kruskal-Wallis is a non-parametric test which determines whether two groups are from a distribution with the same median [62]. Given our orderline of distance we sort the distances and their class membership into separate sets, O_i we also need a corresponding set of ranks, R, where the set of distances in O_i also correspond to R_i. We denote the average rank as $\bar{R}$ and the average rank of the i_{th} class as $\bar{R}_i$.

$$K = \frac{12}{n(n+1)} \sum_{i=1}^{C} |R_i|(\bar{R}_i - \bar{R})^2$$

The final alternative quality measure proposed by [72] is Mood's median. Similar to Kruskal-Wallis, Mood's median is a non parametric test which wants to determine whether two groups are from a distribution with the same

median. Unlike Kruskal-Wallis and the other alternative quality measures, Mood's median does not require the Orderline to be sorted. The measure first starts by creating a contingency table where the counts of each class above or below the median are recorded. It was shown that the median can be found in $O(n)$ time [49]. Where o is the observations, and e is the expected.

$$M = \sum_{i=1}^{C} \sum_{j=2}^{2} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

2.6.3 Data Transformation

The major change from the Shapelet Tree algorithm to the shapelet Transform was the data transformation process. The data transformation is formally defined in algorithm 5. Given a set of a time series **T** and a set of k shapelets **kShapelets** the algorithm calculates the minimum distance between each series and each shapelet. A n x k matrix is constructed from these distance values, in addition to the class values which are appended to the end of the series. The main aim of creating a transformation was to separate the shapelet finding process from the classification process. The main reason for this is that it has been widely shown that there are significantly better classifiers than decision trees and in [47] changing to a support vector machine significantly improved classification accuracy. It was shown that when discriminatory features are not in the time domain it is easier to leverage greater performance than creating more complex classification techniques. It was also shown that transformed data can significantly improve the accuracy of more simple classifiers. It was then shown in [47] that this data transformation was a significant improvement in accuracy on the previous tree-based approaches.

Algorithm 5 $TransformDataset(\mathbf{T}, \mathbf{kShapelets})$

Where **T** is a set of Time Series.
Where **kShapelets** is a set of shapelets.
$n = |\mathbf{T}|$
$k = |\mathbf{kShapelets}|$
Where F is a matrix of size n x k
$i = 1$
for $\mathbf{T}_i$ in **T do**
  $j = 1$
  for S in **kShapelets do**
    F_{ij} =sDist($\mathbf{T}_i, S$)
    $j + j + 1$
return F

2.7 Fast shapelets

Fast shapelets were proposed as a classifier in 2013 [83]. The algorithm is a direct improvement upon the original shapelet selection algorithm and employs a number of techniques to speed up the finding and pruning of shapelet candidates [70]. The major changes made to the shapelet algorithm is the introduction of symbolic aggregate approximation (SAX) [108, 67] as a means for reducing the length of each series as well as smoothing and discretising the data. The other major advantage of using the SAX representation is that shapelet candidates can be pruned by using a collision table metric which highly correlates with Information Gain to reduce the amount of work performed in the quality measure stage.

The FS algorithm is made up of a number of major components. FS embeds the shapelet discovery within a decision tree. The decision tree has been omitted in the algorithmic description in algorithm 6 to improve clarity.

The first stage of the shapelet finding process is to create a list of SAX words [108, 67]. The basic concept of SAX is a two stage process. Firstly, using piece-wise aggregate approximation (PAA) is used to transform a time series into a number of smaller averaged sections, reducing the length and smoothing the series. This aggregated series is then normalized using z

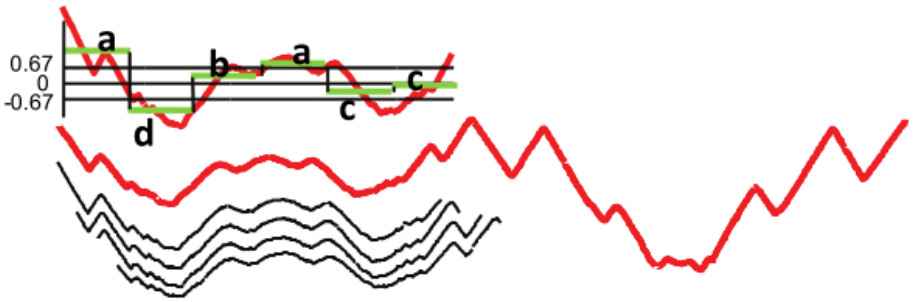


Figure 2.4: Image taken from Fast Shapelets [83].

normalization. With a given alphabet size, in the case of fast shapelets 4, a Gaussian distribution is split into 4 equally likely sections.

$$a < -0.67, -0.67 \geq b < 0, 0 \leq c < 0.67, d > 0.67$$

These four sections discretise the aggregate series into a word. This is shown in Figure 2.4, where part of a series is converted in a SAX word, the figure also demonstrates how a SAX word represents multiple overlapping shapelets because of the aggregation process.

These discretised series are then reduced using random projection, which, given some higher dimension SAX words, reduces their dimensionality by masking a number of letters. The SAX words are randomly projected a number of times, the projected words are hashed and a frequency table for all the SAX words is built [20].

From this frequency table a new set of tables can be built which represent how frequent the SAX word is with respect to all the classes. A score for each SAX word can be calculated based on these grouping scores, and this value is used for assessing the distinguishing power of each SAX word. From this scoring process a list of the top K SAX shapelets can be created. These top K SAX shapelets are transformed back into their original series, where the shapelet quality assessment, which was discussed in further detail in section 2.4, can take place. The best shapelet then forms the splitting rule in the decision tree, identical to the method used in the Shapelet ()Tree.

Algorithm 6 FindBestShapelet(Set of timeseries **T**)

```
1: bsfShapelet, shapelet
2: topK = 10
3: for length ← 5 to m do
4:    SAXList =FindSAXWords(T, length)
5:    RandomProjection(SAXList)
6:    ScoreList =ScoreAllSAX(SAXList)
7:    shapelet =FindBestSAX(ScoreList, SAXList, topK)
8:    if bsfShapelet < shapelet then
9:       bsfShapelet = shapelet
10: return bsfShapelet
```

2.8 Learn Shapelets

Learn shapelets (LS) is an algorithm proposed in [40]. The learn shapelets algorithm is distinctly different from previous shapelet methods in that it does not perform an enumerative search. Learn shapelets uses a gradient descent approach to the shapelet finding problem. A set of initial random shapelets are clustered using k-means. The centroids from these clusters are then refined, using a stochastic gradient descent.

One of the main issues with the learn shapelets method is that the shapelets found are not guaranteed to exist within the training data, and often do not. One of the major benefits of the shapelet tree algorithm, and subsequently the shapelet transform, was that the shapelets are within the data, and provide interpretable features. One of the major reasons for separating the shapelet finding method from the tree based approaches within the shapelet transform was that models built on simpler classifiers were decreasing the performance of shapelets. Learn shapelets classification model is stochastic gradient descent, which is only capable of linearly separating problems.

Algorithm 7 describes learn shapelets. The algorithm begins by finding a number of subsequences in the original training data which require two tuning parameters, defined as R and L. These parameters affect shapelet finding, for example, if we define $R = 3$ and $L = 0.2$ (which are typical

parameters used in the original experiments) we would find shapelets that are 20%, 40% and 60% of the series. The parameters affect the accuracy and the amount of work the algorithm performs. L alters the length of subsequences considered and R affects the coverage of the shapelets, and broadens the search space.

These initial subsequences are then clustered using K-Means in a similar manner to [116]. These subsequence clusters each contain a centroid, which may not be present in the original training data. With the set of centroids a gradient descent model is applied to each. Each shapelet is refined through a defined derivative function, minimizing the entropy loss. This process continues for a max number of iterations, or until the model converges. To increase the success of the learning method, the algorithm has since been refined to use the Adagrad method for on-line learning [31].

Algorithm 7 FindBestShapelet(Set of timeseries $\mathbf{T}$)

```
1: Parameters: K,R,L_min,η,λ
2: S ← InitKMeans(T,K,R,L_min)
3: W ← InitWeights(T,K,R)
4: for i ← maxIter do
5:    M ← updateModel(T,S,α, L_min, R)
6:    L ← updateLoss(T,M,W)
7:    W,S← updateWandS(T,M,W,S, η, R, L_min, L, λ_W, α)
8:    if diverged() then
9:       i = 0
10:      η = η/3
```

2.9 Fused Lasso Generalized eigenvector method

The Fused LAsso Generalised eigenvector method (FLAG) was proposed in [50]. The algorithm is a very recent attempt at optimising the shapelet searching method, by considering methods that have been used in computer vision and bioinformatics. They argue that the shapelet search space is sparse, and as such they can use sparse modelling to find shapelets. In [104] they demonstrate that using a fused lasso function to model the sparsity

of the space, they can also take into account the properties of time series data, because it encourages successive parameter feature estimates to be similar. They demonstrate that using a total-variation regulariser and a ℓ_1 regulariser they make the solution both blocky and sparse. Shapelets tend to exist in groups, that are separated by regions of poor shapelets. By forming a solution that is blocky and sparse the algorithm aims to model this property.

2.10 Random Shapelet Tree and Random Shapelet Forest

The random shapelet tree and subsequent random shapelet forest were proposed in [55, 57, 56]. These methods seek to exploit some of the successes of random forest and in general ensembles of homogeneous classifiers.

The random shapelet tree is a simplistic approach to the shapelet finding problem, but exploits the structure of shapelets in time series. Shapelets tend to be present in clusters of similar quality, both in position and length. A shapelet of length 11, position 2, contains mostly the same values as a shapelet of length 12 in position 3. The definition of a shapelets means that a good shapelet should appear in all the series of the same class. Therefore, the sDist distance value is low for all series of the same class for a given shapelet, and the distance is high for series of other classes. Karlsson et al. [55] demonstrate that randomly selecting shapelets instead of fully enumerating can produce comparable accuracies whilst evaluating a fraction of the search space. citekarlsson16generalized then extend the algorithm to build forests of shapelet trees, in the same manner as a Random Forest, called Random Shapelet Forest (RSF). The data is partitioned into random subsets, which also reduces the the runtime of each tree, as the number of shapelet combinations, and the distance calculations required is much smaller.

2.11 Efficiency Improvements

This section will cover the assorted optimisations that have been proposed for the shapelet tree and shapelet transform in the literature. The shapelet optimizations can be broadly separated into two categories. Either the improvements reduce the average case complexity of the enumerative search by reducing the number of operations performed when evaluating a shapelet candidate or by being able to avoid calculations all together. Alternative improvements reduce the worst case complexity by increasing the worst case memory requirements by caching statistics [83, 79, 70, 38].

2.11.1 Early Distance Abandon and Precomputing

A number of heuristic speed up techniques were proposed [115, 83, 79, 47] to deal with the large volume of calculations required to find the best shapelet. However, even with the speed up techniques proposed shapelet algorithms are still not capable of enumerating the very large datasets, such as StarLightCurves from the UCR-UEA repository [23]. The first speed up technique is relatively simple. The *sDist* function defined earlier in algorithm 3 has a worst case bounding of $O(m^2)$, and is called n times, per shapelet. Whilst the sliding window function is calculating the difference between the current shapelet and the subsequence, the function keeps track of the smallest distance found so far. Whilst comparing the two subsequences, the algorithm is calculating the sum of the individual positions in both respective series. If the partial square sum becomes greater than the square of the smallest distance found so far, that particular series cannot be a good match and the distance calculation can be early abandoned. This early abandon technique was explained in the original paper [114], and is demonstrated in Figure 2.5. Ideally finding good matches to a shapelet early in the *sDist* function, the amount of work that can be avoid is potentially very large. Early abandon techniques have been shown that whilst they do not reduce the overall worst case time complexity of an algorithm they are still very effective in reducing the average case runtime [114]. Part of the

contributions in this thesis (see chapter 4) improves upon this technique and so it is pertinent to describe it in detail here.

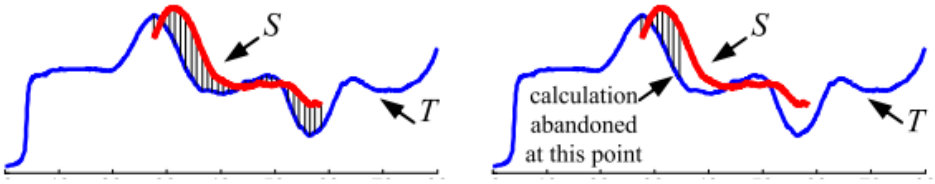


Figure 2.5: Early Abandon of a time series (T) and a shapelet (S) being compared using the *sDist* function. In the illustration on the left, S an T are pairwise compared using Euclidean distance. In the diagram on the right, S and T are compared using Euclidean distance which has an early abandon point illustrated. The diagram is taken from [115]

Mueen et al. [79] proposed the caching of summary statistics to offset the large time requirements for calculating the distance of a shapelet to a series, the technique makes the trade off of memory in favour of speed, and reduces the run time complexity of the distance function from O(m^2) to constant time. Each shapelet and the subsequences it is compared with during the distance calculation need to be length-normalized, using z normalization, this is to ensure that differences in scale and offset do not affect any similarity in shape [59]. Given two series A and B of length m The normalised Euclidean distance is calculated by:

$$nDist(A,B) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\left(\frac{A_i - \bar{A}}{\sigma_A}\right) - \left(\frac{B_i - \bar{B}}{\sigma_B}\right)\right)^2} \tag{2.3}$$

To calculate this normalised Euclidean distance requires $O(m)$ time, Mueen proposed that with 5 sufficient statistics it is possible to calculate the distance in constant time [92].

Given two series A and B the main five statistics are the sum of values and the squared sum of values, for each series, and the pairwise sum of products of the two series, which are presented in Equation 2.4. The mean

and variance for each series can be calculated simply from these statistics, which are shown in Equations 2.5 and 2.6.

$$\sum_{i=1}^{m} A_i, \sum_{i=1}^{m} B_i, \sum_{i=1}^{m} A_i^2, \sum_{i=1}^{m} B_i^2, \sum_{i=1}^{m} A_i B_i \tag{2.4}$$

$$\bar{A} = \frac{1}{m} \sum A \tag{2.5}$$

$$\sigma_A^2 = \frac{1}{m} \sum A^2 - \bar{A}^2 \tag{2.6}$$

With these statistics positive correlation and normalised subsequence distance can be calculated (shown in Equation 2.7).

$$C(A, B) = \left(\frac{\sum_{i=1}^{m} A_i B_i - m\bar{A}\bar{B}}{m\sigma_A \sigma_B} \right) \tag{2.7}$$

$$dist(A, B) = \sqrt{2(1 - C(A, B))} \tag{2.8}$$

When calculating the orderline for any single shapelet a large proportion of the calculations overlap and there is unnecessary redundancy. Given two time series A and B, any length and starting position in A is considered a potential shapelet, a number of the calculations for overlapping Euclidean distance calculations could be used, but because the subsequences are zNormalised this is not immediately possible. The sum of products($\mathbb{S}_A$, $\mathbb{S}_B$) and the sum of products squared($\mathbb{S}_A^2$, $\mathbb{S}_B^2$) are recorded, and a final Matrix is constructed which stores each configuration of the sum of products for the subsequences in A and B ($\mathbb{M}$). These arrays are indexed using the positions for A and B, and so depending on which sets of statistics are required, the distance calculation can be extracted. The *sDist* function is redefined using this methodology in algorithm 8.

In algorithm 8 the distance calculation is performed according to the redefined normalise distance. The cached statistics are extracted as the loop iterates for each position in the single time series.

$$\bar{A} = \frac{\mathbb{S}_A[u + l - 1] - \mathbb{S}_A[v - 1]}{l}$$

Algorithm 8 $sDistCached(u, l, Stats_{A,B})$

```
min = ∞
for v ← 1 to |B| − |A| + 1 do
  dist = √(2(1 − C(A, B)))
  if dist ≤ min then
    min = dist
return min
```

$$\bar{B} = \frac{\mathbb{S}_B[u+l-1] - \mathbb{S}_B[v-1]}{l}$$

$$\sigma_A = \frac{\mathbb{S}_A^2[u+l-1] - \mathbb{S}_A^2[v-1]}{l} - \bar{A}^2$$

$$\sigma_B = \frac{\mathbb{S}_B^2[u+l-1] - \mathbb{S}_B^2[v-1]}{l} - \bar{B}^2$$

2.11.2 Entropy Pruning

The second speed up technique proposed in [114] was early entropy pruning. The distance between the shapelet and all other series is calculated to form the orderline, which is used in the information gain calculation. Instead of calculating the whole set of distances required, which is one of the most expensive operations in the shapelet algorithm, an upper bound for the information gain is calculated. The information gain is calculated as each new distance value is calculated in $sDist$, the data series that have not been compared with the current shapelet are placed on the orderline in the ideal position that would maximise entropy. In Figure 2.6 a complete set of series is calculated where for a single shapelet nm^2 operations have been performed. In Figure 2.7 four of the distances have been calculated, and four have been placed in the best configuration.

As each distance is calculated the orderline places any distances not calculated in the optimal position. Series which are the same class as the shapelet are placed on the far left (0 distance) and the other class is placed on the right. The entropy pruning calculates an upper bound on the information

gain, which can be compared to the information gain of the best shapelet found so far. If the upper bound would not result in a change of best shapelet found, then the algorithm can early abandon calculating entire series as even in the best case scenario, the current shapelet would be worse. In the case of the shapelets in the Shapelet Transform, the last shapelet in the kShapelets list is used as the entropy pruning threshold. Part of the contributions in this thesis (see chapter 4) improve upon the concept of this technique and so it is pertinent to describe it in detail here.

Figure 2.6: Fully calculated orderline with two classes.

Figure 2.7: Partially calculated orderline with two classes. The series that have not been calculated are placed in best case positions.

2.11.3 Similar shapelet abandon

Mueen et al. [79] also proposed a novel pruning technique in [79]. Given a shapelet $S_{i,l}$ they ask the question, "how good can $S_{i+1,l}$ be?". In the special case where the distance from the first shapelet to the second shapelet is 0.

$$dist(S_{i,l}, S_{i+1,l}) = 0$$

In this case we know that the information gain for the second shapelet is also going to be identical, and so it can be pruned. In a more realistic scenario the

first and second shapelets will be very similar. If the second shapelets quality can be identified, then it would be possible to generate an upper bound. Mueen defines the distance between two shapelets as $dist(S_{i,l}, S_{i+1,l}) = R$ and that the sDist between a time series is $sDist(S_{i+1,l}, T_j)$, he suggests that by triangular equality $sDist$ can be as low as $sDist(S_{i+1,l}, T_j) - R$ and as high as $sDist(S_{i+1,l}, T_j) + R$. By this reasoning, the next shapelet in the series has a range of quality of $-R$ to $+R$ from its current position. They describe a set of operations with a given orderline for the shapelet $S_{i,l}$ the best case movement of the distances on the orderline for $S_{i+1,l}$ such that an upper bound on its information gain can be calculated. With this upper bound we can then decide whether we want to commit to the much more costly process of evaluating its true Information Gain.

2.12 Shapelet Search improvements

There have been a number of algorithms that have sought to improve the process of finding shapelets, some of the most common algorithms were described in detail in previous sections [40, 83, 55, 88, 38, 87]. Boosting and bagging has been applied to random shapelet forests to increase performance [57, 56]. Minor improvements to the learn shapelets algorithm has been shown with in [41]. Some of the work on both learn shapelets, and random shapelet forests has also been applied in the multivariate domain, however with only a few datasets publicly available experimental analysis is minimal.

2.13 Timing Experiments

To demonstrate the current run times for the most common shapelet algorithms in the literature, seven of the smallest datasets were chosen from the UCR-UEA repository. The results are present in the Table 2.1. In this particular set of experiments, the machine used was a raspberry pi 2. This machine is not designed to run machine learning algorithms in any optimised way. However, with a very lightweight operating system, and controlled environment the timings should be reasonably unbiased. Despite a low-power

machine, the results will all be relative to each other and should give an approximate understanding of the speed of each of the four algorithms. The table demonstrates the speed of the Fast Shapelets algorithm, and is in-line with the claims made in the literature. It was expected that the Shapelet Tree would be the slowest, and that Fast Shapelets would be fastest. With the Shapelet Transform marginally slower than Learn Shapelets. The timings presented were calculated as the average over five runs.

Dataset	ST	LS	FS	STree
CBF	3706568	1000960	**45850**	6651483
ECGFiveDays	2719910	292173	**16722**	3518036
ItalyPowerDemand	43051	46596	**1304**	135659
MoteStrain	326837	103908	**5436**	438783
SonyAIBORobotSurface2	229204	90518	**4992**	314740
SonyAIBORobotSurface	166140	66158	**4142**	225397
TwoLeadECG	395843	111239	**3807**	530001

Table 2.1: Timing Results for ST, LS FS and STree in milliseconds.

2.14 Applications of Shapelets

The shapelet approach to time series classification has been applied to numerous problems within the research community. Within the UEA group they have been used on electric device classification, classification of mutant worms and classifying hand outlines [47, 71, 70]. In the original paper the algorithm was applied to leaf outlines [115], further application of the Shapelet Tree algorithm includes gesture recognition [45] and gait recognition [96, 114]. In both of these instances marked improvements were seen from other approaches. In [83] the Fast Shapelets algorithm was used on the outline of horned lizards and turtle skulls, classifying the species and demonstrating the interpretable nature of shapelets on outline problems.

For most of the literature, shapelets or similar motif finding algorithms have been designed and applied to univariate data. The problem of how to handle shapelets in a multivariate domain is an interesting challenge, both in

terms of minimizing workload and producing accurate interpretable results. McGovern et al. [76] applied a similar technique to shapelets on multivariate tornado data, attempting to predict weather patterns. Ghalwash et al. [35] applied shapelets to handle multivariate diagnostic data, which was used for early predictions. The key problems they encountered were dealing with phase independent features across the dimensions which remains an open problem [34].

Shaplet based learning exists outside of classification where examples they have been used in clustering [47, 116, 105] and similar concepts to shapelets were explored in early classification [112, 113, 42, 14].

2.15 Issues With Current Approaches

There a number of issues with the current approaches to shapelet finding and classification. The problems with the shapelet tree were identified and the shapelet transform was proposed as a way of mitigating the issues with embedding the shapelet discovery in a decision tree [70]. There are still a number of issues with the Shapelet Transform, however, which extend to all enumerative shapelet methods. The first major problem is multi class information gain is not very effective at separating one shapelet well from the rest, this is discussed in greater detail chapter 4. The Shapelet Transform still enumerates the entire problem space and on very large datasets such as StarlightCurves full enumeration is still untenable. Logical Shapelets and Fast Shapelets both embed the shapelet discovery and rule implementation in a decision tree. It was shown in [70] the shapelet tree method is significantly worse than a transform based approach and so by extension these methods could benefit greatly from being separate from the classification process. The SAX method for reducing series length in Fast Shapelets smooths the series with PAA. This has the effect of smoothing a series and potentially removing some fundamental shapes within the series. Learn shapelets generates centroids from some initial shapelets and updates them in an online method. The shapelets that are generated as a result of this are often not present in the original dataset, and so lose some of

there interpretability, the runtime is unpredictable because of the learning process. If the algorithm cannot converge a restart and new random shapelets begins the process again. This also means that the memory footprint can be variable.

2.16 Multivariate Time Series Classification

Multivariate time series classification (MTSC) has been gaining traction within the research community. The major issue, until recently, with multivariate time series analysis is that as the length of the series and the number of dimensions increase they become increasingly difficult to analyse in realistic time frames. Whilst this problem may not have been directly solved, as computing power has increased, the ability to work on larger datasets and more complex problems has become easier. Paired with the fact that internet of things (IoT) devices and smart devices are increasingly more common, it means that this type of data is being collected more widely.

One of the major areas of research within MTSC is activity and gesture recognition, otherwise known as human activity recognition (HAR). Gesture and activity recognition is the problem of recognising a particular movement or action within a time series, where the class defining action is potentially phase independent, and the signal to noise ratio is often quite high. The user is potentially performing many different actions, only one of which we are trying to detect. One of the difficulties in activity recognition is that the users are potentially already in motion or performing actions that are potentially similar, for example, some of the classes in UWaveGesture (down-up/up-down)(clockwise-circle/anti-clockwise circle) [73]. One of the other difficulties with gesture recognition is that the potential features could exist in one dimension, in all, or some. For example, given a simple hand gesture that is recorded by tracking X, Y, and Z movement. The type of hand gesture may move only through the X and Z planes, with no movement in the Y. Another example of the same movement will have the same shapelet in the X and Z, but if the Y channel is noisy this could be a source of difficulty. Identifying phase inter-independence and intra-dependence is a

difficult problem, and identifying which dimensions are noisy and which contain the signal is also a very difficult challenge when working in the multivariate domain.

Despite these potential problems gesture recognition has become one of the most popular areas of research [103, 64, 32, 52, 61, 58]. Musical instrument activity recognition is an extension of the general activity recognition problem and some interesting research has been conducted in this area [36, 102].

Multivariate time series data does not consist solely of activity and movement based data. There has been a large amount of research into the health domain, specifically electroencephalogram (EEG) classification, or balance and mobility sensor data for patients with Parkinsons disease (PD) [77, 39, 1]. EEG classification is a potentially interesting area of research, with test subjects looking at different images on a computer screen, and classifying based on the electrical signals. EEG classification forms part of human-computer interaction research, with research focusing on whether meaningful signals, or input, can be data mined from these multivariate time series.

Some other areas of research have extended to handwriting classification [10], similarity between image textures [28] and mining of historical manuscripts [117].

Most of these research domains have focused on using dynamic time warping with a nearest neighbour classifier, mainly because until very recently it was considered the state of the art solution to time series classification [86]. Specialized approaches to multivariate time series classification include adaptive dynamic time warping, dependent dynamic time warping, and independent dynamic time warping algorithms which are covered in greater detail in section 2.17 [98, 99].

Two-dimensional singular value decomposition was proposed as an unsupervised approach to MTSC, where the covariance matrix of the samples is formed, and the row-row and column-column features are extracted and used in a 1-nearest-neighbour classifier [110]. Two-dimensional locality preserving projections were also proposed where the MTSC samples are projected into a lower dimensional space and the class features are closer to each other and

a 1-nearest-neighbour classifier can be used [109].

One of the major criticisms of recent multivariate time series classification, is that the algorithms are tested on a handful of datasets, which are not shared across the research community. Comparing algorithms is very difficult, and quantifying the improvements over different approaches has not been performed yet [13]. The other major criticism of time series classification in general is that source code is often not publicly available and verifying experiments is not possible. One of the aims of this thesis is to unify the problem set for multivariate time series classification and provide a framework to prove advancements in the field. In chapter 6 we discuss in detail the contributions made to MTSC and the datasets.

2.17 Multivariate Dynamic Time Warping

Three forms of multivariate dynamic time warping have been proposed recently [98, 24, 99]. The dynamic time warping algorithm is modified to consider two different types of multivariate similarity. These types of features are considered independent and dependent of the dimension. Dependent dynamic time warping (DTW_D) was proposed for use in historical text mining [24] and independent dynamic time warping (DTW_I) was later proposed in addition to adaptive dynamic time warping (DTW_A) in [98, 99]. DTW_A is a combination of the two distances with a novel selection criteria. DTW_A is designed to be in the worst case no worse than the better of the two distance measures on any particular problem. In this section we will explain the specifics of multivariate dynamic time warping, and the selection criteria for which method is used on a particular series in DTW_A.

DTW_D and DTW_I are very simple modifications to the DTW algorithm. Given two dimensional multivariate time series, Q and C which have two dimensions X and Y. Dependent dynamic time warping finds the shortest path when combining distances inside the warping window. Independent dynamic time warping, calculates individual distances for each series and each dimension, and then combines the distances. This is formally defined in Equation 2.9 and Equation 2.10. Adaptive dynamic time warping (DTW_A)

was defined as way to dynamically select which multivariate version of DTW was best suited to the dataset and demonstrated that in the worst-case DTW_A was no worse than either of its components.

$$DTW_D(Q, C) = DTW(Q_X, Q_Y, C_X, C_Y) \tag{2.9}$$

$$DTW_I(Q, C) = DTW(Q_X, C_X) + DTW(Q_Y, C_Y) \tag{2.10}$$

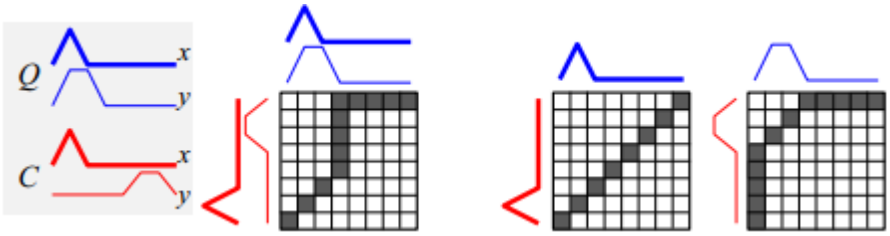


Figure 2.8: A simple diagram of a two dimensional time series comparison using Independent and Dependent dynamic time warping. The image on the (left) is DTW_D and the image on the (right) is DTW_I. Image taken from [99]

2.18 Multivariate Shapelet Algorithms

In section 2.10 the random shapelet forest algorithm was described. This method has been expanded to consider multivariate time series classification. The series are treated as independent time series and the forest is built by splitting the MTSC into separate dimensions [80, 53, 54].

In section 2.8 the Learn Shapelets algorithm is described in detail. Learn Shapelets was extended to search for single dimension shapelets which are selected randomly from both the series, length and position. These shapelets are then tuned in the same manner as the univariate algorithm [41].

Ghalwash et al. [35] showed that extraction of shapelets in multivariate time series data could be used for early prediction[34]. Multivariate shapelet detection (MSD) was proposed. The algorithm extracts multiple shapelets

and calculates the information gain for each, these multivariate shapelets are weighted using a modified information gain that prioritise earlier found shapelets in the series.

Chapter 3

Experimental Methodology

In this chapter we present the datasets used for experimentally comparing the changes made to the Shapelet Transform. In this thesis we compare many classifiers across multiple datasets. The datasets we use have been standardised as a set of 85 datasets, that are forever expanding [66, 23]. The 85 datasets were standardised as a joint effort between the University of East Anglia (UEA) and the the University of California Riverside (UCR). The aim of having a standardised set of problems is that it makes comparing classifiers more robust. Until recently most classifiers were compared using an arbitrary number of datasets that were compared via simple win/loss counts.

3.1 Comparing Classifiers

To test this thesis more thoroughly we use a statistically rigorous test for multiple classifiers across many datasets. This procedure was first outlined in [29] and is designed to test for statistical significance between classifiers. The test is based on a two-stage rank-sum test using the non-parametric analysis of variance (ANOVA).

The first stage of the approach is to test the null hypothesis against the alternative hypothesis. The null hypothesis is that there is no difference between the average ranks of c classifiers on d datasets. The alternative

hypothesis is that at least one classifier's mean rank is different.

M is a c by d matrix of classification accuracies, where $M_{i,j}$ is the accuracy of the i^{th} dataset on the j^{th} classifier. M is formally defined in Equation 3.1.

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1c} \\ m_{21} & m_{22} & \dots & m_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ m_{d1} & m_{d2} & \dots & m_{dc} \end{bmatrix} \tag{3.1}$$

The next stage is to calculate a c by d matrix R which contains the ranks of the classifiers where $r_{i,j}$ is the rank i^{th} dataset on the j^{th} classifier and the ranks of equal classifiers are averaged. We formally define R in Equation 3.2.

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1d} \\ r_{21} & r_{22} & \dots & r_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ r_{c1} & r_{c2} & \dots & r_{cd} \end{bmatrix} \tag{3.2}$$

From the matrix R the average rank for a single classifier j is calculated by $\bar{r}_j = \frac{\sum_{j=1}^{d} r_j}{d}$. To test the hypothesis the Friedman statistic F is calculated using Equation 3.3. This is an estimate using a Chi-squared distribution with (c-1) degrees of freedom. This tests whether there is a difference in the mean ranks of any of the classifiers.

$$\chi^2 = \frac{12d}{c(c+1)} . \left[\sum_{j=1}^{c} \bar{r}_j^2 - \frac{c(c+1)^2}{4} \right] \tag{3.3}$$

Demšar [29] note that χ^2 is considered conservative and so proposed using Equation 3.4 which follows the F distribution.

$$F = \frac{(d-1)\chi^2}{d(c-1) - \chi^2} \tag{3.4}$$

The F distribution has (c-1) and (c-1)(d-1) degrees of freedom under the null hypothesis. If the null hypothesis can be rejected, and one of the classifiers has an average rank that is significantly different to any of the

others, the second stage of the test begins [29]. Demšar [29] perform pair-wise Nemenyi tests to find the differences between the classifiers. The test for determining whether two classifiers are significantly different is known as the *critical difference*, which is presented in Equation 3.5.

$$CD = {}^{q_a}\sqrt{\frac{c(c+1)}{6d}} \tag{3.5}$$

Where q_a is calculated by the difference in the range of standard deviations from the smallest valued sample, and the largest valued sample. Demšar [29] suggest that, by comparing classifiers in this way, a diagram showing the differences between the rankings and the significances can be shown. Classifiers that are no significantly different from one another are shown in *cliques*. These cliques are represented by black bars. In Figure 3.1 we present an example critical difference diagram.

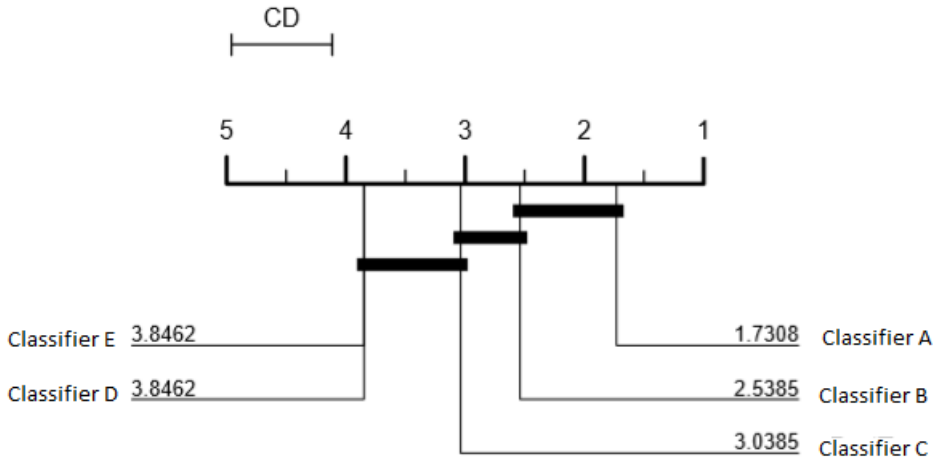


Figure 3.1: An example Critical Difference (CD) diagram demonstrating how to interpret the results from a pairwise comparison of five classifiers over multiple datasets.

Figure 3.1 demonstrates five classifier, A,B,C,D and E. To interpret the results of this critical difference is straightforward and demonstrates their utility. Classifier A and B are in the same clique, and are not significantly

different. However, classifier A is significantly different to C,D, and E. Classifier B is not significantly worse than A or classifier C, but is significantly better than D and E. Classifier C is significantly worse than classifier A, but is not significantly better than B,D and E. Finally D and E are no significantly worse than each other or classifier C, but are significantly worse than A and B. This diagram provides a good breakdown of the rankings of each classifier and how they compare to one another. The *cliques* give an understanding of where classifiers are similar in performance and where one or more classifiers may be better than others.

3.2 Performance Statistics

When using critical difference diagrams we often compare by the error rate, but we sometimes refer to accuracy as well, which we define as $(1 - error)$.

In this section we will define the performance measure we use throughout this thesis when comparing classifiers. Most commonly we compare by error, but critical difference diagrams can be formed with any of the statistics we present in this section. Comparing classifiers by these additional statistics can reveal different improvements that are more subtle.

We define a dataset D as a set of attribute vectors which are paired with a class variable, $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, where n is the number of instances within the dataset and the set of class labels is: $y \in \{1, ..., C\}$.

Classification is the mapping from the space of possible attributes to the class labels, where we derive a probability distribution over all the values of the class. This distribution is defined as: $\hat{\mathbf{p}}$. Given the i^{th} instance in the dataset, the probability distribution is $\hat{\mathbf{p}}_i = \{\hat{p}_i(y = 1|\mathbf{x}_i), ..., \hat{p}_i(y = C|\mathbf{x}_i)\}$. Given the distribution, the class value is defined as the maximum probability in the distribution (the most likely). So for the i^{th} instance in a dataset we derive its class label by:

$$\hat{y}_i = \underset{j=1,..C}{\operatorname{argmax}} \hat{p}(j) \tag{3.6}$$

A correctness function is defined as $f(y, \hat{y})$. If the prediction is correct

then it will return a 1, if incorrect it will return 0.

$$f(y, \hat{y}) = \begin{cases} 1, & \text{if } y = \hat{y}. \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

The error is simply calculated by the number of incorrect class labels predicted, for example if we have 100 cases to predict, and we calculate the probability distributions for all of them, if we guess 8 incorrectly the error rate is 0.08 and the accuracy is 0.92.

Sometimes we want to calculate the error with respect to the classes. We define this as the balanced error. This metric accommodates for class imbalances in the dataset. To calculate balanced error we calculate the proportion of each class that is correct and then calculate the sum of the proportional classes with respect to the class distributions. In Equation 3.8 we define d_j where the j^{th} class value is the proportion of correct classes recorded.

$$d_j = \frac{\sum_{y_i \in D, y_i = j} f(y, \hat{y})}{\sum_{y_i \in D} f(y, j)} \tag{3.8}$$

The proportion of class j in the dataset is defined as e_j. The balanced error is calculated in Equation 3.9

$$\sum_{j=1}^{C} d_j \cdot e_j \tag{3.9}$$

3.3 Standard Classification Algorithms

Throughout this thesis we use a number of standard classification algorithms, often as part of the Heteregenous Ensemble of Standard Classification Algorithms (HESCA) (see section 4.5). In addition to HESCA we also use these algorithms in chapter 6 for benchmarking on multivariate datasets is performed where these standard algorithms are used on concatenated data series and on dimensional ensembles.

Though it may seem unintuitive to use standard classification algorithms on time series data, a large volume of research has been conducted on these standard algorithms. This is especially poignant when we consider the shapelet transform. Some of the best algorithms in the literature [8] use data transformation to transform time series into other domains, where more standard classification algorithms are potentially more effective.

3.3.1 C4.5 Decision Tree

The C4.5 (also known as J48) is a decision tree classifier which was first discussed in [82]. The algorithm has been widely used in the literature [33, 65]. Decision trees aim to partition the training data by selecting effective splitting rules. The criteria for a good splitting rule is that it can separate classes well. The shapelet tree (see section 2.4) was designed as a form of decision tree. C4.5 uses gain ratio and information gain (see subsection 2.4.2 to decide the quality of a split, where the aim is to maximise separation of the classes. The algorithm is a greedy top down approach where the data is partitioned around these best splits and the data is split between the two subsequent nodes created, this process continues recursively splitting the data until either a maximum depth is reached, or only one class remains in that partition. The tree is pruned after this process to avoid over-fitting by removing nodes.

3.3.2 Support Vector Machine

Cortes and Vapnik [26] first introduced Support Vectors Machines(SVM) in 1995. SVMs have been used extensively in general machine learning. The simplest type of SVM uses a linear kernel, where it is assumed the data is linearly separable. Given a dataset $\mathbf{T}$ with two classes and given $C = \{1, -1\}$ the SVM aims to build the function $f(\mathbf{T})$ so that:

$$f(\mathbf{T}) = \begin{cases} c^* = +1, & \text{if} \geq 0 \\ c^* = -1, & \text{if} < 0 \end{cases} \tag{3.10}$$

where $f(\mathbf{T})$ is of the form:

$$f(\mathbf{T}) = w.\mathbf{T} + b \tag{3.11}$$

w is the weight vector to $f(\mathbf{T})$ and b is the bias that offsets the weight vector. The objective of the SVM is too find the best vector that can separate the two classes. The SVM iteratively updates this vector with respect to the data until the all the classes are separated from each other. The problem with this particular form of fitting is that it is prone to over fitting on the train data. More complex SVMs aim to find the maximum margin between classes. Other types of optimisations allow for some misclassification during the training stage which can also reduce the complexity of the polynomial, and increase generalisation.

For many problems a linear separation is not a good model, and they require more complex hyperplanes to partition the data. Non-linear kernels can be specified and in all of our experiments that involve HESCA (see section 4.5) we use a quadratic kernel as part of the ensemble. However, as the kernel becomes increasingly more complex fitting the planes becomes more computationally expensive, and in addition fitting higher order polynomials can result in over fitting as well. These are some of the reasons why throughout this work we only use linear and quadratic SVMs.

3.3.3 Random Forest

In addition to using heterogeneous ensembles we also consider homogeneous ensembles. Random forest is one such algorithm that builds an ensemble of decision tree classifiers [19]. One of the main problems with decision trees is that they can be prone to over-fitting. Random forest seeks to solve some of these issues by artificially creating diversity in the train data. The data set is partitioned into many different subsets by removing different attributes. A forest is initialised with a fixed number of trees, this is usually 500 trees. Each of these trees is assigned a random subset of the original data, where a decision tree is built on this particular partition. Decision trees have been covered in great detail in this thesis, both in terms of partitioning classes

using information gain and how the trees are constructed. In chapter 2 we also described the random shapelet forest algorithm which is inspired by Random Forest.

3.3.4 Rotation Forest

Rotation Forest is similar to random forest in many respects [89]. Rotation forest initially selects subsets of the training data, similarly to random forest. Each of these subsets is then transformed using principle component analysis (PCA). PCA is used to transform the data into an alternative representation, the principle components of which are used train the C4.5 classifier. Both Rotation Forest and Random Forest use majority voting schemes to create the predictions.

3.4 Resampling Datasets

In chapter 4 we identified that consistency of datasets between testing of classifiers in the literature had room for improvement. The UCR-UEA repository was created as a large collection of datasets to test algorithms on. One of the major problems with standardised datasets is that the train and test splits are often arbitrarily created and potentially even made artificially harder than the problem type should be for particular families of classifiers.

To solve some of these issues we wanted to develop a more rigorous experimental methodology that the time series classification community could adopt. In [8] we proposed stochastically resampling the datasets with respect to the original datasets distribution. This ensures that the original train and test sizes were maintained and the problems time and space requirements were respected. This means that if in the original splitting of the dataset key features happened to be in the test set, over a larger sample size the accuracy of classifiers on those datasets should increase, with the original split being an outlier. Given the 85 datasets, we chose to resample each problem 100 times, where we refer to a particular resample as a fold. Fold 0 is always the original problem and folds 1-99 are resamples where the

random seed for the stochastic sampling is the fold number. By ensuring that fold 0 is the original dataset it means that all of our newly generated results are comparable with work that does not use this methodology.

The major problem with this method is that instead of 85 datasets, each classifier is evaluated on 8500 problems, which drastically increases the amount of experimental work required to compare algorithms.

Multivariate problems are becoming increasingly popular within the literature we wanted to extend this methodology to the multivariate domain as well which we discuss in chapter 6.

3.5 Univariate Datasets

In this chapter we introduce the 85 UCR-UEA datasets [4, 23], this archive was recently increased from originally 41 datasets to the new 85 following a large scale experimental evaluation [8]. A summary of the types of problems can be seen in Table 3.1 and in Tables 3.2, 3.3, 3.4, 3.5, 3.6, 3.7 and 3.8 the break down of the individual problems and there type is shown.

The datasets can be broadly separated into seven categories. These are image outline; sensor reading; motion capture; spectographs; ECG measurements; electric devices and simulated datasets. In Table 3.1 we detail the number of problems in each type.

Image Outline	29
Sensor Reading	16
Motion Capture	14
Spectographs	7
ECG measurements	7
Electric Devices	6
Simulated	6
Total	85

Table 3.1: Number of datasets by problem type

In this section, each dataset will be presented in tables organised by problem type. These tables will outline the properties of each dataset and the relevant sizes of the training data. These will include training and test

size, series length and the number of classes. The first table is Electric Device based datasets presented in Table 3.2. The dataset Electric Devices contains the largest train size of any of the datasets in the UCR-UEA repository. These types of problem are captured using smart meter devices in homes, where electricity usage is monitored over different periods of time. These problems tend to be classification of the device based on electricity usage.

Name	Train size	Test size	Length	Num. Classes
Computers	250	250	720	2
ElectricDevices	8926	7711	96	7
LargeKitchenAppliances	375	375	720	3
RefrigerationDevices	375	375	720	3
ScreenType	375	375	720	3
SmallKitchenAppliances	375	375	720	3

Table 3.2: Electric Device Datasets

The next type of problem is electrocardiogram (ECG) type problems shown in Table 3.3. These types of problem are the signals generated from heartbeat monitoring equipment, the problems include classification of different patients, or trying to detect different heart defects that may or may not be present in the signal.

Name	Train size	Test size	Length	Num. Classes
CinCECGtorso	40	1380	1639	4
ECG200	100	100	96	2
ECG5000	500	4500	140	5
ECGFiveDays	23	861	136	2
NonInvasiveFetalECGThorax1	1800	1965	750	42
NonInvasiveFetalECGThorax2	1800	1965	750	42
TwoLeadECG	23	1139	82	2

Table 3.3: ECG Datasets

The image based datasets are the largest group of any problem type in the archive (shown in Table 3.4). These types of problem are extracted by calculating the Euclidean distance between the centre of the image and outline identified. Transforming the outline into a 1D signal (shown in Figure 3.2). A time series is not only temporal data, but can be any data that is ordered sequentially.

Figure 3.2: An example outline image created converted into a time series.

Name	Train size	Test size	Length	Num. Classes
Adiac	390	391	176	37
ArrowHead	36	175	251	3
BeetleFly	20	20	512	2
BirdChicken	20	20	512	2
DiatomSizeReduction	16	306	345	4
DistalPhalanxOutlineAgeGroup	400	139	80	3
DistalPhalanxOutlineCorrect	600	276	80	2
DistalPhalanxTW	400	139	80	6
FaceAll	560	1690	131	14
FaceFour	24	88	350	4
FacesUCR	200	2050	131	14
FiftyWords	450	455	270	50
Fish	175	175	463	7
HandOutlines	1000	370	2709	2
Herring	64	64	512	2
MedicalImages	381	760	99	10
MiddlePhalanxOutlineAgeGroup	400	154	80	3
MiddlePhalanxOutlineCorrect	600	291	80	2
MiddlePhalanxTW	399	154	80	6
OSULeaf	200	242	427	6
PhalangesOutlinesCorrect	1800	858	80	2
ProximalPhalanxOutlineAgeGroup	400	205	80	3
ProximalPhalanxOutlineCorrect	600	291	80	2
ProximalPhalanxTW	400	205	80	6
ShapesAll	600	600	512	60
SwedishLeaf	500	625	128	15
Symbols	25	995	398	6
WordSynonyms	267	638	270	25
Yoga	300	3000	426	2

Table 3.4: Image Datasets

Motion datasets tend to be collected from gyroscope or accelerometer recording devices. These signals are multivariate time series consisting of X, Y and Z components of the movement. The Cricket and UWaveGestureLibrary datasets are covered in more detail in the multivariate datasets section. Until recently many of the algorithms have solely focused on univariate datasets and therefore these multivariate datasets are separated into multiple univariate

problems, in the case of Cricket and UWaveGesture the dimension is denoted at the end of the datasets name. In the case of UWaveGestureLibraryAll, this is a concatenation of the three dimensions in the order X,Y,Z.

Name	Train size	Test size	Length	Num. Classes
CricketX	390	390	300	12
CricketY	390	390	300	12
CricketZ	390	390	300	12
GunPoint	50	150	150	2
Haptics	155	308	1092	5
InlineSkate	100	550	1882	7
ToeSegmentation1	40	228	277	2
ToeSegmentation2	36	130	343	2
UWaveGestureLibraryAll	896	3582	945	8
UWaveGestureLibraryX	896	3582	315	8
UWaveGestureLibraryY	896	3582	315	8
UWaveGestureLibraryZ	896	3582	315	8
Worms	181	77	900	5
WormsTwoClass	181	77	900	2

Table 3.5: Motion Datasets

Sensor readings are a typical application of time series classification (Table 3.6). In these datasets sensors are capturing particular types of information. In the case of StarlightCurves the dataset consists of many star light curves, where these are the brightness of a celestial object as a function of time. StarlightCurves is one of the largest datasets in the repository. The aim of chapter 5 is to be able to evaluate this dataset in a reasonable time frame.

Name	Train size	Test size	Length	Num. Classes
Car	60	60	577	4
Earthquakes	322	139	512	2
FordA	3601	1320	500	2
FordB	3636	810	500	2
InsectWingbeatSound	220	1980	256	11
ItalyPowerDemand	67	1029	24	2
Lightning2	60	61	637	2
Lightning7	70	73	319	7
MoteStrain	20	1252	84	2
Phoneme	214	1896	1024	39
Plane	105	105	144	7
SonyAIBORobotSurface1	20	601	70	2
SonyAIBORobotSurface2	27	953	65	2
StarlightCurves	1000	8236	1024	3
Trace	100	100	275	4
Wafer	1000	6164	152	2

Table 3.6: Sensor Datasets

Bagnall et al. [7] described a simple simulator framework. As part of [8] many time series classification algorithms were taxonomised into families that find certain types of features. These taxonomies motivate the need for creating datasets that can artificially prove the effectiveness of classifiers on the particular types of problems they are aiming to solve. Furthermore, with simulated datasets, the signal to noise ratio can be increased to see how certain families of algorithm perform as the data becomes more noisy. Shapelets are described as phase independent subsequences, the simulator creates different phase independent patterns for the number of classes specified, these are then inserted randomly into time series. Noise is then applied to the simulated datasets and we can then show that Shapelets should be the best algorithm on this type of problem. In some preliminary experiments and testing with sampling methods we were able to use simulated datasets to check the correctness of search functions as part of the software testing process. This is because the location of the subsequences were known.

Name	Train size	Test size	Length	Num. Classes
Beef	30	30	470	5
Coffee	28	28	286	2
Ham	109	105	431	2
Meat	60	60	448	3
OliveOil	30	30	570	4
Strawberry	613	370	235	2
Wine	57	54	234	2

Table 3.8: Spectograph Datasets

Problem size	Counts	Proportion
1-100	31	36.47
101-500	34	40.00
501-1000	14	16.47
>1000	6	7.06

Table 3.9: Distribution of Problem sizes

Name	Train size	Test size	Length	Num. Classes
CBF	30	900	128	3
ChlorineConcentration	467	3840	166	3
Mallat	55	2345	1024	8
ShapeletSim	20	180	500	2
SyntheticControl	300	300	60	6
TwoPatterns	1000	4000	128	4

Table 3.7: Simulated Datasets

The final set of datasets are spectograph problems (see Table 3.8). A spectrograph is a machine that separates light into a frequency signal. These problems tend to be food based, in the case of Meat, Ham and Beef the problems are aimed at detecting whether the particular meat is fraudulent.

Finally, in Table 3.9 the sizes of the problems based on the length of the time series is presented. In the case of shapelets the length of time series can have a large effect on the runtime of the algorithm.

3.6 Multivariate Datasets

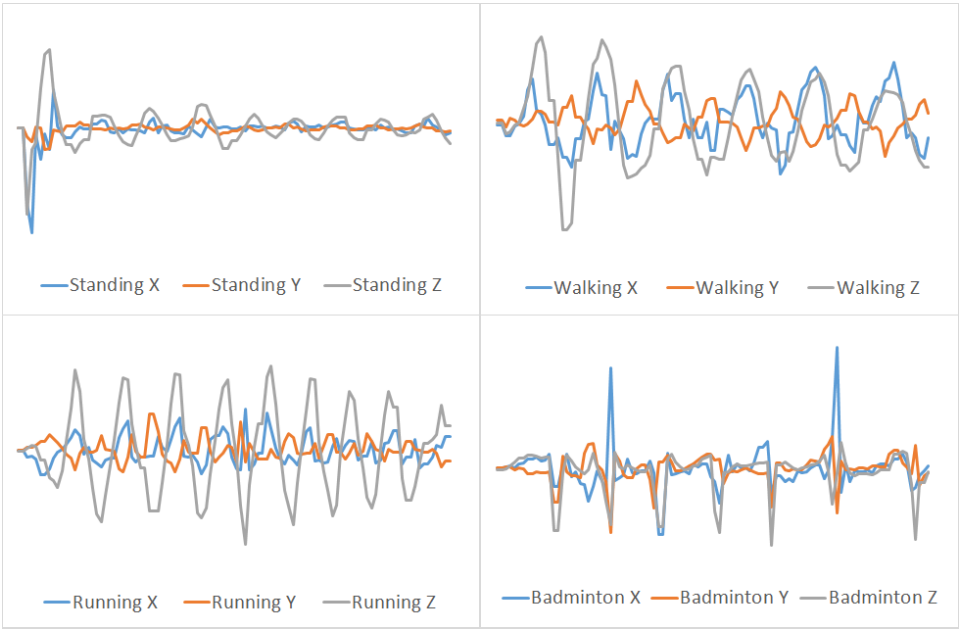
To evaluate new multivariate methods, and to benchmark against other algorithms from the literature, a set of multivariate datasets were created

and collated from the literature which span a range of different problem types. In Figure 3.4 a list of the datasets and there respective properties are shown. The datasets have a range of different sizes, number of instances, length of the series, the number of series and finally number of classes. One caveat on the current datasets, is that to simplify and reduce the need for extensive individual dataset knowledge when benchmarking we have reduced some problems into sub problems. This is most notable with the AALTD problems. These were originally from a challenge dataset produced for the ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (AALTD). The original aim being to classify six different gestures using eight spatial sensors placed on an individual, resulting in 3 dimensional movement information for each sensor. We opted to split the dataset into a separate classification problem for each sensor. However, this will have the effect of artificially making the problems more difficult as class discriminating information may not be contained in a particular sensory dataset, or could be across multiple sensors.

The aim of this work was to unify the published multivariate datasets under a common framework using the ARFF format in Weka [43]. The following datasets were extracted and converted: AALTD; ArabicDigit [44]; Japanese vowels [63]; Cricket, Handwriting, ArticularyWord [99]; PEMS [27]; PenDigits [2]; UWaveGesture [73]; Epilepsy [106].

The final dataset is MVMotion. There are three variants: MVMotionA; MVMotionG; and MVMotionAG. Data was collected from a 3D accelerometer and a 3D gyroscope on a mobile device during a particular set of activities. The general type of problem is Human Activity Recognition (HAR) and is similar in concept to the Epilepsy dataset. All MVMotion datasets consist of four classes, which are walking, resting, running and badminton. Participants were required to record motion a total of five times, and the data is sampled once every tenth of a second, for a ten second period. We demonstrate an example of each of the classes, for accelerometer data in Figure 3.3a and for the gyroscope data in Figure 3.3b. The datasets are constructed, MVMotionA is X,Y,Z accelerometer data, MVMotionG is X,Y,Z gyroscope data, and MVMotionAG is both, forming a six dimensional problem.

(a) Accelerometer MVMotion datasets



(b) Gyroscope MVMotion dataset

Figure 3.3: An example of the four classes for both Accelerometer data from the MVMotion dataset.

datasets	n	d	m	c
AALTD_0	90	3	52	6
AALTD_1	90	3	52	6
AALTD_2	90	3	52	6
AALTD_3	90	3	52	6
AALTD_4	90	3	52	6
AALTD_5	90	3	52	6
AALTD_6	90	3	52	6
AALTD_7	90	3	52	6
ArabicDigit	6599	13	94	10
AWordLL	275	3	145	25
AWordT1	275	3	145	25
AWordUL	275	3	145	25
CricketLeft	84	3	1198	12
CricketRight	84	3	1198	12
HandwritingA	150	3	153	26
HandwritingG	500	3	153	26
JapaneseVowels	270	12	30	9
MVMotionA	40	3	101	4
MVMotionAG	40	6	101	4
MVMotionG	40	3	101	4
PEMS	267	144	964	7
PenDigits	7494	2	9	10
UWaveGesture	120	3	316	8
VillarData	137	3	207	4

Figure 3.4: A list of the datasets in the multivariate time series archive. Number of instances is denoted by n, number of dimensions is denoted by d, length of series is denoted by m, and number of classes is denoted by c

.

Chapter 4

Improving the accuracy and reducing the runtime of the Shapelet Transform

Contributing Publications

- A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with COTE: The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27:2522–2535, 2015
- A. Bostrom and A. Bagnall. Binary shapelet transform for multiclass time series classification. *Proc. 17th International Conference on Big Data Analytics and Knowledge Discovery (DAWAK)*, 2015
- A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advance. *Data Mining and Knowledge Discovery*, pages 1–55, 2016
- A. Bostrom and A. Bagnall. Binary shapelet transform for multiclass time series classification. *Transactions on Large-Scale Data and Knowl-*

edge Centered Systems XXXII: Special Issue on Big Data Analytics and Knowledge Discovery, pages 24–46, 2017

4.1 Introduction

The shapelet transform is an approach to time series classification that finds and extracts phase independent subsequences. The main algorithm is covered extensively in chapter 2. The number of shapelets that exist within a single time series is calculated with respect to the length, m, and the minimum length and the maximum length. For a single series it has $\sum_{l=3}^{m}(m-l+1)$ shapelets, which is simplified to $\frac{m^2-3m+2}{2}$. The worst case runtime complexity for finding shapelets is therefore bounded by m^2 for a single series. When the number of series is defined as n, examining all shapelets is $O(nm^2)$. To evaluate a single shapelet, the algorithm slides the shapelet along each series in the dataset performing normalisation and Euclidean distance calculations on these subsequences. This process is also $O(nm^2)$ as the algorithm essentially compares each shapelet to every other shapelet. Overall the runtime of this algorithm is bounded by $O(n^2m^4)$ which as m and n grow large can become impractical to search in a reasonable time frame.

The UCR-UEA datasets are a set of standardised problems which are covered in chapter 3 in more detail. One of the largest problems in this list is StarLightCurves. The attributes for StarlightCurves are $m = 1024$, $n = 1000$. Currently this problem cannot be enumerated. When the shapelet length parameters are set to $min = 3$ and $max = 1024$, the number of total shapelets to evaluate are $522,753,000$. The approximate number of operations overall is in the order of 10^{18}.

The distance calculation techniques presented in the original shapelet paper [114] and subsequent improvements presented in [79, 70, 84] have all sought to reduce the average runtime of the algorithms that evaluate time series and shapelets. The main work in this chapter improves upon these heuristic methods and improves the accuracy of the shapelet transform

when used on multi-class classification problems. These results have been presented in [15, 16]. Finally we perform a thorough analysis of the current best shapelet algorithms on a large problem set to establish a benchmark for future work which was presented in [8].

4.2 Comparison of Published Results

There are three main shapelet algorithms in the literature. These three algorithms are; the Shapelet Transform (ST) [47], Fast Shapelets (FS) [83] and Learn Shapelets (LS) [40]. One of the major problems we have identified when comparing and evaluating to algorithms in the literature is that not all have been tested on a common set of problems. For example the published results for Learn Shapelets was evaluated on 45 datasets, Fast Shapelets on 33, and the Shapelet Transform on 75. In Table 4.1 we present the data for these three algorithms on the intersection of the datasets. We present the critical difference diagram in Figure 4.1 where we show that both the Shapelet Transform and Learn Shapelets are not significantly worse than each other, but are both significantly better than Fast Shapelets.

Datasets	LS	FS	ST
Adiac	0.5632	0.4859	0.6777
Beef	0.76	0.5533	0.9
CBF	0.9944	0.9471	0.9822
ChlorineConcentration	0.6514	0.5831	0.6893
Coffee	1	0.9321	1
DiatomSizeReduction	0.9667	0.883	0.8824
ECGFiveDays	1	0.9959	0.993
FaceAll	0.7825	0.5893	0.7544
FaceFour	0.9522	0.9102	0.9091
FacesUCR	0.9413	0.6717	0.9151
Fish	0.9337	0.8028	0.9942
GunPoint	1	0.9393	1
ItalyPowerDemand	0.9695	0.905	0.9582
Lightning2	0.823	0.7049	0.6557
Lightning7	0.8027	0.5972	0.69863
Mallat	0.9543	0.9672	0.9126
MedicalImages	0.7295	0.567	0.6632
MoteStrain	0.9129	0.783	0.8794
OliveOil	0.44	0.7867	0.9333
SonyAIBORobotSurface1	0.8974	0.6855	0.9434
SonyAIBORobotSurface2	0.9184	0.7852	0.8919
SwedishLeaf	0.913	0.7307	0.9376
Symbols	0.9644	0.9324	0.9266
SyntheticControl	0.9927	0.919	0.9967
Trace	1	0.998	1
TwoLeadECG	0.9974	0.9097	0.9912

Table 4.1: Published Results for LS, FS and ST

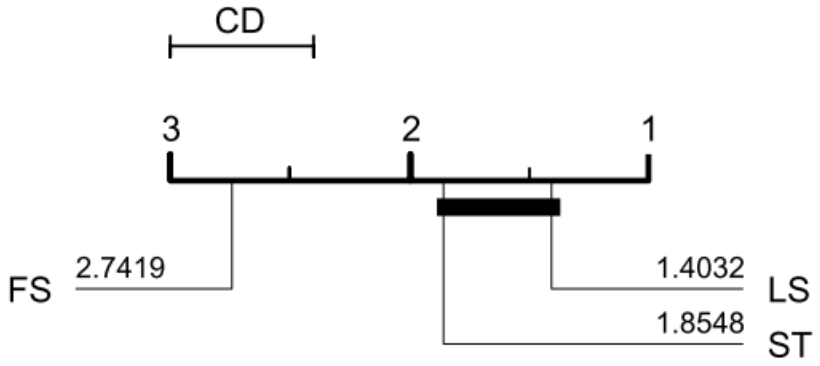


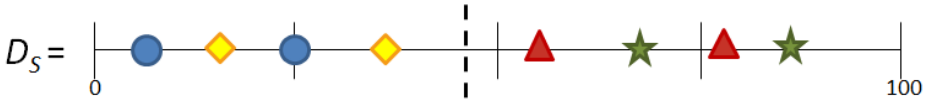
Figure 4.1: Critical difference of published results from Table 4.1

4.3 Multi-class information gain

In chapter 2, four shapelet quality measures were discussed. These were: f-stat; information gain; Mood's median; and Kruskal Wallis. In particular information gain was discussed in detail in section 2.4. All three of the shapelet based algorithms identified use information gain to assess the quality of shapelets. However, one of the major problems with information gain on many class problems is that useful information about a single class can be lost. Figure 4.2 presents two multi-class orderlines used in the Shapelet Transform, each of which is evaluating a single shapelet (see subsection 2.4.1). These two orderlines demonstrate the major problem with information gain on multi-class problems. They highlight how the way in which the algorithm currently extracts and evaluates the best shapelets may be incorrect, and could lead to worse accuracy on the transform. As the number of classes in a dataset increase it becomes more likely that a single class is difficult to single out. In the case of Fast Shapelets and the Shapelet Tree algorithm, this was not a problem as the set of time series that shapelets were extracted from is reduced at each new depth of the tree. One of the other major problems with shapelet finding is that if one class dominates the problem space, and subsequently produces many high quality shapelets, they could dominate the transform. Whilst overall there would be good accuracy on one class the transform may perform poorly on other less represented classes. Datasets that are highly imbalanced will have a naturally high baseline when using a naive majority vote classifier (zero rule classifier). To mitigate this problem the number of shapelets found (k) was set very large ($10n$) and it was left to the classifier to be able to deal with redundant features. However, this can lead to very large transforms as the transform becomes a nx$10n$ matrix. The hypothesis is that by balancing the number of shapelets for each class, the number of shapelets that are required in the final transform could be reduced. Overall this should increase the classification accuracy on multi-class problems, and enable a reduction in the size of k such that the classification time is reduced.

To improve multi-class classification we create a one-versus-all encoding

(a) orderline 1



(b) orderline 2

Figure 4.2: An example orderline split for two shapelets. Orderline (a) discriminates between class 1 and the rest, however orderline (b) has the higher information gain.

for the shapelets when constructing the orderline, which in the case of Figure 4.2 would result in orderline 1 (a) having the higher information gain of the two. As a by product of creating a two-class classification problem we can leverage the early abandon entropy pruning presented in the original shapelet paper on these other problems and therefore reduce the average runtime.

In algorithm 9 we present the updated shapelet transform that includes class balancing and the binary shapelets. The main changes are that on lines 1-3 we create a map of the best shapelets for each class, which is evenly distributed among all classes, where the value k is the number of total shapelets to keep, each class is given a proportion of this as k/C. In line 11 we collect the shapelets into a list, and in line 12 they are sorted by their quality. In lines 14-16 the best quality shapelets are merged with those of there respective class and stored in a map, where the class value is used as the key. In the "findDistances" function a set of distances is produced, and paired with an associated class value from each series. In the case of the binary class changes that have been proposed this class value is either the same class as the shapelet (0) or not the same (1).

Algorithm 9 BinaryShapeletSelection(**T**, min, max, k)

Input: A list of time series **T**, min and max length shapelet to search for and k,the maximum number of shapelets to find)

Output: A list of k Shapelets

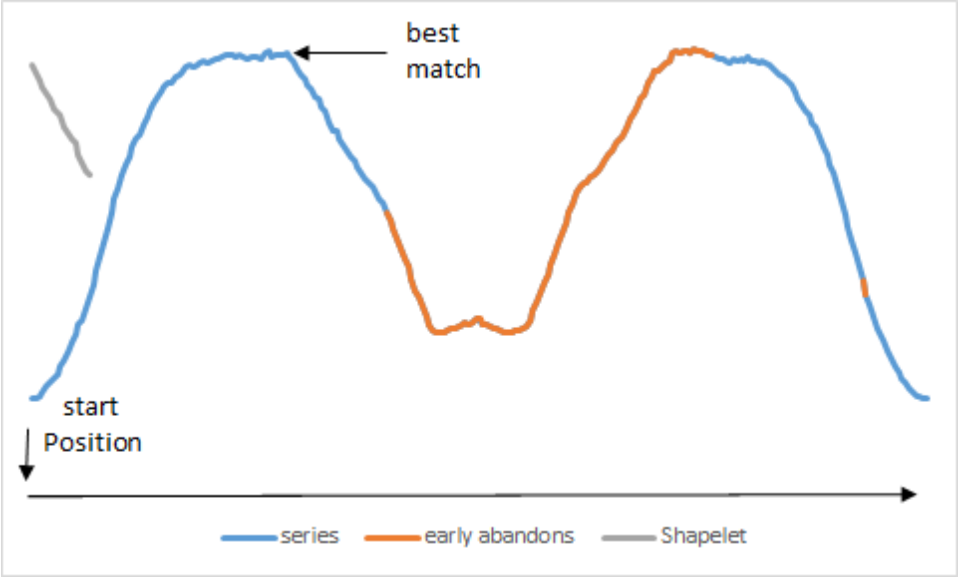
1: $numClasses \leftarrow$ getClasses(**T**)
2: $kShapeletsMap \leftarrow \emptyset$
3: $prop \leftarrow k/numClasses$
4: **for all** T_i in **T do**
5: $shapelets \leftarrow \emptyset$
6: **for** $l \leftarrow min$ to max **do**
7: $W_{i,l} \leftarrow generateCandidates(T_i, l)$
8: **for all** subseries S in $W_{i,l}$ **do**
9: $D_S \leftarrow findDistances(S,\mathbf{T})$
10: $quality \leftarrow assessCandidate(S, D_S)$
11: $shapelets.add(S, quality)$
12: $sortByQuality(shapelets)$
13: $removeSelfSimilar(shapelets)$
14: $kShapelets \leftarrow kShapeletsMap.get(T.class)$
15: $kShapelets \leftarrow merge(prop, kShapelets, shapelets)$
16: $kShapeletsMap.add(kShapelets, T.class)$
17: **return** $kShapeletsMap.asList()$

4.4 Changing the shapelet evaluation order

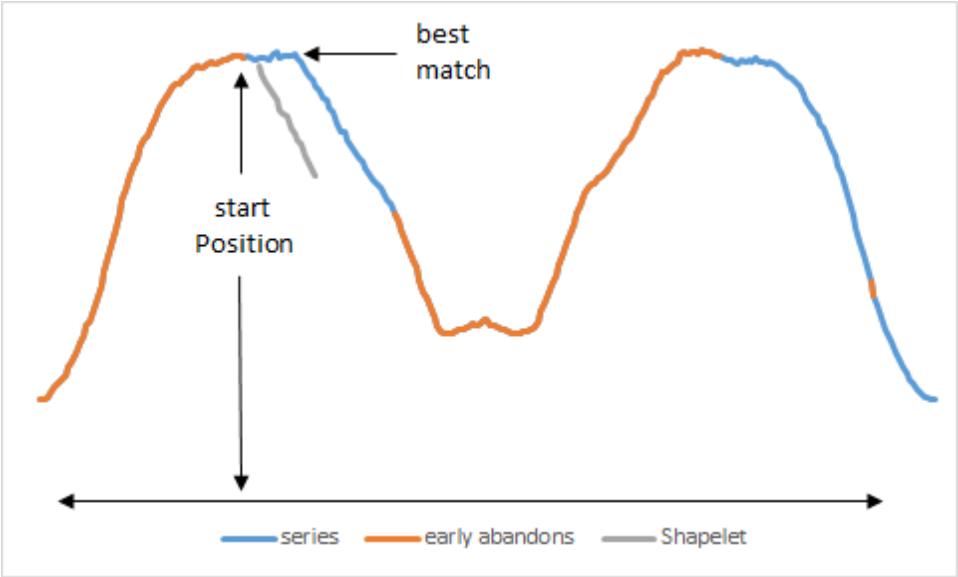
Having introduced our proposed binary classification of shapelets we were able to leverage entropy pruning (see subsection 2.11.2) because of the two class classification on multiclass problems. In addition to this speed up, we wanted to further reduce the number of calculations performed in the sliding window distance function ($sDist$). We propose a reordering of the sliding window function.

Instead of taking the shapelet of length l and sliding it along from position 0 to position $m - l + 1$, the new method starts from the position in which the shapelet is found and then by sliding left and right until the left reaches 0, and the right reaches $m - l + 1$. Figure 4.3 illustrates the difference between these two distance measures. In the case of the reordered distance measure, on average more early abandons take place during the distance calculations. In the worst case, which is where no early abandoning takes place, this algorithm will perform no more operations than the current. In algorithm 10 the algorithm is given a shapelet S and a time series T. Given the start position of the shapelet and its length a subsequence is extracted and normalised. While the algorithm can still traverse left or right in the sliding window function the loop continues. The current position is checked to see if it will exceed either the left most element of the array (position 0) or the right most position ($m-l+1$). If either the left or right positions are able to be evaluated this distance calculation is performed, alternating left and right. These left and right distances are compared with the best distance found so far to enable early abandoning. Alternating subsequences are continually evaluated until both a left traversal and a right traversal are no longer possible, as the start position rarely occurs exactly in the middle of a series, the left will finish before the right or vice-versa. The algorithm presented is greatly simplified to illustrate the traversal process. To ensure parity with other distance measures summary statistics are maintained when traversing both left and right. This means that online normalization can be performed in constant time. Maintaining summary statistics for bi-directional traversal means in the case of a right traversal the left most element is subtracted

from the statistics and the new right most element is added. In the case of a left traversal the opposite is true, the right most element is subtracted and the left most element is added. In addition to these summary statistics, the initial shapelet is sorted by the size of the values in the array. Corresponding indexes are used when performing the euclidean distance calculations to further increase the number of early abandons possible, this method was presented in [84].



(a) current early abandon



(b) proposed early abandon

Figure 4.3: An example of Euclidean distance early abandon where the *sDist* scan starts from the beginning (a) and from the place of origin of the candidate shapelet (b).

Algorithm 10 $sDist($ shapelet S,series $T_i)$

```
1: subSeq ← getSubSeq(T_i, S.startPos, S.length)
2: bestDist ← euclideanDistance(subSeq, S)
3: i ← 1
4: while leftExists || rightExists do
5:    leftExists ← S.startPos − i ≥ 0
6:    rightExists ← S.startPos + i ≤ T_i.length − S.length + 1
7:    if rightExists then
8:       subSeq ← getSubSeq(T_i, S.startPos + i, S.length)
9:       currentDist ← earlyAbandonDistance(subSeq, S, bestDist)
10:      if currentDist > bestDist then
11:         bestDist ← currentDist
12:   if leftExists then
13:      subSeq ← getSubSeq(T_i, S.startPos − i, S.length)
14:      currentDist ← earlyAbandonDistance(subSeq, S, bestDist)
15:      if currentDist > bestDist then
16:         bestDist ← currentDist
17:   i ← i + 1
18: return bestDist
```

4.5 Heterogeneous ensemble of standard classification algorithms

The previous shapelet transform was used in conjunction with the Weighted Ensemble (WE) [46]. The weighted ensemble is a set of simple classification algorithms formed into a cross-validated weighted ensemble specifically designed for use with the Shapelet Transform. Initially the weighted ensemble comprised of the classifiers used to evaluate the transform in the original paper. These classifiers were; C4.5, 1NN, Naive Bayes, Bayesian Network, Rotation Forest, Random Forest, Support Vector Machine (Linear), and Support Vector Machine (Quadratic). Since the inception, the weighted ensemble has been refined and the results have been presented in [65]. The weighted ensemble was subsequently changed to be called the heterogeneous ensemble of simple classification algorithms (HESCA), and includes five classifiers. The algorithms that make up HESCA are: a support vector

machine with a polynomial(linear) kernel [81]; a multi-layer perceptron [90]; logistic regression; nearest neighbour with euclidean distance; and a C4.5 decision tree. These five classifiers are deliberately not tuned, as one of the ideas behind HESCA is that it is easy to leverage off of the diversity of classifiers that are similar in performance, but drastically different in design. It was shown in [65] that the heterogeneous ensemble, which includes an untuned SVM, outperforms on average a computationally expensive tuned SVM. We formally define HESCA as follows, given a set of k classifiers $M = \{M_1, ..., M_k\}$, and the unseen case $\mathbf{x}$. Each of these classifiers will produce a probability distribution, which must be combined to form the final ensemble's probability distribution $\hat{p_k}(\mathbf{x})$. HESCA employs a simple exponentially weighted majority vote over the probability distributions of each classifier. Training consists of defining the weighting of each classifier, for which we simply use the estimated accuracy found through cross-validation of the training data.

$$\hat{p}(y = i|M, x) \propto \sum_{j=1}^{k} w_j^{\alpha} p_j(y = i|M, \mathbf{x}) \tag{4.1}$$

As well as accuracy, one could quite easily use any other performance metric calculable from a cross validation: balanced accuracy; log likelihood; area under the receiving operator characteristic; f-score. These have all been implemented. We opted to use accuracy for both its simplicity to calculate and motivate, and the fact that in experimentation, it was no worse than any of the rest. The α parameter used in the exponentiation is designed to accentuate differences in the classifier's weightings. When $\alpha = 0$, all weightings become equal, while as α tends to ∞, the ensemble becomes functionally equivalent to the strongest classifier found through cross validation. α is somewhat arbitrarily set to 4, to avoid a parameter search.

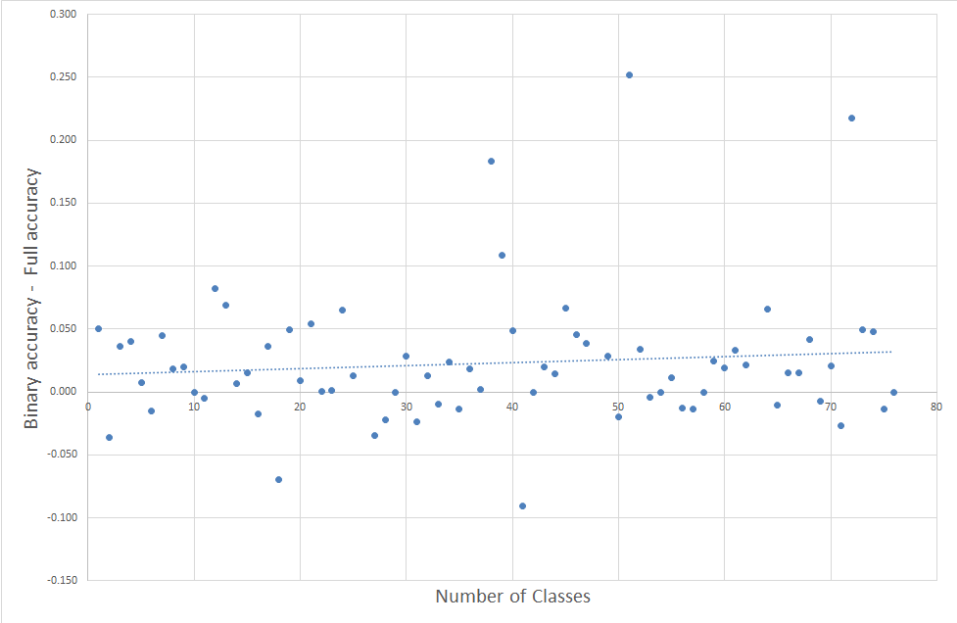


Figure 4.4: Number of classes plotted against the difference in error between the full shapelets and the binary shapelets. A positive number indicates the binary shapelets are better. The dotted line is the least squares regression line.

4.6 Results

In Table 4.2 we present the wins and losses for the Balanced Shapelet Transform compared with the published results for the Shapelet Transform on 75 datasets [70]. In addition to this, the full table of results is presented in the appendix (Table 2). Figure 4.4 presents the regression plot of the difference in errors between them on the y axis and the number of classes on the x. The plot shows a minor trend that the difference in error rate between them increases as the number of classes increases.

Number of classes	Full Better	Binary Better
2 class	6	19
3-5 class	7	13
6-9 class	4	8
>10 class	4	8
All	21	48

Table 4.2: Number of data sets the binary shapelet beats the full shapelet split by number of classes.

4.7 Analysing the individual Improvements

In Table 4.2 we show that the Binary Shapelet Transform is more effective than the original Shapelet Transform on multiclass problems. To gain insight into the changes we made to the algorithm we separated the transform into four distinct transforms to compare individual sections. Initially we test the *min* and *max* shapelet length heuristic proposed in the original paper (paramST). We also tested for the default parameters of $min = 3$ and $max = m$(ST), we then tested including binary shapelets (binST), and finally we added balancing onto the transform (BST).

The shapelet transform is deterministic in nature, however HESCA is not. We perform the accuracy experiments 30 times to collect the mean and standard deviation. We opted to use the 40 smallest datasets from the UCR-UEA archive. We defined the smallest datasets by computing a value based on the shapelets worst case operation count.

In Figure 4.5 we show on the four algorithms that there is no significant difference between the different approaches. Out of all 40 datasets eighteen of them are two class problems, for which we think balancing could decrease accuracy. The binary shapelets without balancing has the lowest of the four ranks, which suggests that without balancing binary shapelets the overall accuracy can become worse. The multi class wins and losses are presented in Table 4.2. On problems with two classes, the balancing and binary shapelets are not as effective as ST, but they are not significantly worse. On large multi class problems the Binary and balancing, makes the Shapelet Transform win

more often when compared with the original ST.

We propose creating a single classifier called ST_HESCA which depending on the number of class labels either enables binary labels and class balancing or disables them. Class balancing and binary shapelets are not required on two class problems and in some cases the balancing can hinder overall accuracy. In the two class case, one class dominating the transform can be more effective than dividing the shapelets evenly, and choosing potentially lower quality shapelets from the other class. Irrelevant of whether the binary and class balancing are enabled or disabled the transform will still use HESCA as its base classifier.

dataSets	numClasses	paramST	ST	binST	BST
ArrowHead	3	0.766	**0.778**	0.766	0.777
Beef	5	0.833	**0.9**	0.767	0.833
BeetleFly	2	0.9	**0.95**	0.9	0.9
BirdChicken	2	0.7	0.8	0.85	**0.85**
CBF	3	**0.996**	**0.996**	0.968	0.952
Coffee	2	0.964	**1.0**	**1.0**	**1.0**
DiatomSizeReduction	4	**0.922**	0.866	0.903	0.899
DistalPhalanxOutlineCorrect	2	0.784	0.758	0.741	**0.792**
DistalPhalanxOutlineAgeGroup	3	0.765	**0.796**	0.786	0.787
DistalPhalanxTW	6	0.635	**0.68**	0.647	0.679
ECG200	2		**0.84**	0.836	0.828
ECGFiveDays	2	0.997	0.994	0.997	0.997
FaceAll	14	0.728	0.762	0.775	**0.779**
FaceFour	4	**1.0**	0.886	0.764	0.783
FacesUCR	14	**0.92**	0.889	0.904	0.919
GunPoint	2	**1.0**	**1.0**	**1.0**	**1.0**
ItalyPowerDemand	2	0.95	**0.953**	0.95	0.948
Lightning7	7	**0.767**	0.711	0.703	0.699
MedicalImages	10	0.615	0.64	0.66	**0.681**
MiddlePhalanxOutlineCorrect	2	0.599	**0.626**	0.591	0.584
MiddlePhalanxOutlineAgeGroup	3	0.739	**0.789**	0.78	0.775
MiddlePhalanxTW	6	**0.569**	0.551	0.538	0.563
MoteStrain	2	0.887	0.947	0.95	**0.951**
OliveOil	4	**0.933**	0.9	0.8	0.833
Plane	7	**1.0**	**1.0**	**1.0**	**1.0**
ProximalPhalanxOutlineCorrect	2	**0.858**	0.835	0.834	0.834
ProximalPhalanxOutlineAgeGroup	3	**0.921**	0.911	0.893	0.904
ProximalPhalanxTW	6	0.8	**0.834**	0.82	0.821
ShapeletSim	2	0.994	**1.0**	**1.0**	**1.0**
SonyAIBORobotSurface1	2	**0.938**	0.835	0.827	0.822
SonyAIBORobotSurface2	2	0.892	**0.952**	0.951	0.94
SwedishLeaf	15	0.916	0.926	0.931	**0.937**
Symbols	6	**0.939**	0.909	0.895	0.893
SyntheticControl	6	0.967	0.968	0.985	**0.987**
ToeSegmentation1	2	0.956	0.965	0.969	**0.982**
ToeSegmentation2	2	0.838	0.938	0.938	**0.946**
Trace	4	**1.0**	0.98	**1.0**	0.99
TwoLeadECG	2	**1.0**	**1.0**	**1.0**	**1.0**
Wine	2		**0.87**	0.815	0.832
Wins		10	12	0	9

Table 4.3: Table of the accuracies for the 4 variations of the shapelet algorithm, classified using HESCA

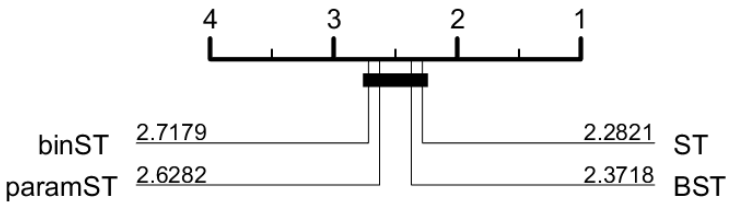


Figure 4.5: The critical difference diagram of Table 4.3

4.8 Measuring heuristic speed up techniques

Our primary aim in this work was to create a shapelet transform that could handle multi class problems better than the full shapelet transform. In section 4.6 we demonstrated the improvements that binary shapelets and class balancing could achieve. In algorithm 10 we described a new heuristic speed up measure for calculating the distances of a shapelet to the time series in the dataset. This sliding window distance function is defined as $sDist$ in the algorithmic description of the shapelet transform in chapter 2. To measure the efficacy of these heuristic changes we need to define a measure for evaluating the operations performed by a single transform. We define this value as the calculation of the Euclidean distance between a shapelet and subsequence in a time series. In Equation 4.2 we formally define the

Algorithm Name	Parameters
paramFST	min and max set via length heuristic
FST	$min = 3, max = m$
Prune	FST and entropy pruning
RoundRobin	Prune and round robin ordering
Online	RoundRobin and sorting of shapelet indexes by value
ImpOnline	Online and algorithm 10
binFST	ImpOnline and binary shapelets
BST	binFST and algorithm 9

Table 4.4: A table of the seven different parameters used to measure the reduction in number of operations performed by the shapelet transform

equation for calculating the number of fundamental operations in a shapelet transform, where n is the number of time series, m is there length, and min and max are the constraints on the shapelets to evaluate.

$$opCount = n \sum_{l=min}^{max} (m - l + 1)^2 l(n - 1) \tag{4.2}$$

We design seven experiments to record the number of operations each transform makes on a set of the same 40 datasets we previously used in Table 4.3. We defined 7 sets of parameters for the transforms, and because the transform is deterministic and we are not measuring accuracy these experiments were only performed once.

The paramFST test will use the shapelet heuristic for setting min and max parameters, and no other optimisations. The prune test will use the heuristic pruning of series based on the best case projected information gain. The RoundRobin test will build upon the pruning test parameter set, by additionally alternating between series of different classes to see if evaluation order. The Online test will build upon the parameter set of RoundRobin by using the heuristic early abandon techniques by sorting the shapelet indices before the sDist function, as well as caching and updating summary statistics. The ImpOnline test will build upon the Online test by using the updated sDist proposed in algorithm 10. BinFST uses the same parameter

set as ImpOnline but will enable binary classification on multi-class problems, without balancing the classes. Finally the BST test is binFST parameter set with the class balancing enabled from algorithm 9.

datasets	ST	prune	RoundRobin	Online	ImpOnline	binFST	BST	paramST
ArrowHead	1	0.99	0.98	0.50	0.31	0.30	0.30	0.43
Beef	1	0.68	0.63	0.36	0.23	0.27	0.28	0.02
BeetleFly	1	0.91	0.91	0.58	0.55	0.55	0.56	0.16
BirdChicken	1	0.90	0.90	0.47	0.40	0.40	0.40	0.16
CBF	1	1	1	0.77	0.73	0.73	0.73	0.48
Coffee	1	1.00	1.00	0.45	0.09	0.09	0.09	0.04
DiatomSizeReduction	1	1	1.00	0.36	0.09	0.09	0.09	0.01
DistalPhalanxOutlineAgeGroup	1	1	1	0.47	0.19	0.19	0.19	0.47
DistalPhalanxOutlineCorrect	1	1	1	0.49	0.22	0.22	0.22	0.16
DistalPhalanxTW	1	1	1	0.47	0.19	0.19	0.18	0.32
ECGFiveDays	1	1	1	0.58	0.33	0.33	0.33	0.63
FaceAll	1	1	1	0.72	0.69	0.69	0.69	0.27
FaceFour	1	1.00	1.00	0.69	0.56	0.57	0.57	0.41
FacesUCR	1	1	1	0.74	0.68	0.68	0.67	0.56
GunPoint	1	1.00	1.00	0.59	0.36	0.36	0.36	0.35
ItalyPowerDemand	1	1	1	0.53	0.40	0.40	0.40	0.55
Lightning7	1	1.00	1.00	0.73	0.71	0.71	0.70	0.24
MedicalImages	1	1	1	0.64	0.62	0.62	0.53	0.41
MiddlePhalanxOutlineAgeGroup	1	1	1	0.46	0.16	0.16	0.15	0.46
MiddlePhalanxOutlineCorrect	1	1	1	0.46	0.16	0.16	0.16	0.10
MiddlePhalanxTW	1	1	1	0.45	0.16	0.16	0.16	0.47
MoteStrain	1	1	1.00	0.69	0.58	0.58	0.58	0.32
OliveOil	1	0.59	0.65	0.28	0.01	0.01	0.01	0.01
Plane	1	1	1	0.57	0.45	0.45	0.45	0.88
ProximalPhalanxOutlineAgeGroup	1	1	1	0.44	0.14	0.14	0.14	0.47
ProximalPhalanxOutlineCorrect	1	1	1	0.45	0.14	0.14	0.14	0.10
ProximalPhalanxTW	1	1	1	0.44	0.14	0.14	0.14	0.45
ShapeletSim	1	0.81	0.80	0.69	0.67	0.67	0.68	0.01
SonyAIBORobotSurface1	1	1	1	0.58	0.40	0.40	0.40	0.52
SonyAIBORobotSurface2	1	1	1	0.65	0.56	0.56	0.56	0.60
SwedishLeaf	1	1	1	0.55	0.39	0.39	0.39	0.41
Symbols	1	1	1	0.63	0.60	0.60	0.60	0.42
SyntheticControl	1	1	1	0.77	0.77	0.77	0.77	0.62
ToeSegmentation1	1	0.98	0.98	0.62	0.61	0.61	0.61	0.67
ToeSegmentation2	1	0.94	0.95	0.55	0.56	0.56	0.55	0.60
Trace	1	1	1	0.69	0.63	0.63	0.63	0.77
TwoLeadECG	1	1	1	0.50	0.20	0.20	0.20	0.09

Table 4.5: A Table showing the percentage of operations performed for each of the 7 parameter sets which are compared to a complete exhaustive search without optimisations.

In Table 4.5 we show the percentages for operations performed in the sDist function for the 7 shapelet variations described earlier. The averages as a portion of the amount of work done are shown in Table 4.6. This shows the decreasing amount of work required for each stage of the improvements.

Stats	FST	Prune	RoundRobin	Online	ImpOnline	binST	BST	paramST
Avg.	1	0.968	0.968	0.554	0.391	0.392	0.389	0.370
Std.	0	0.086	0.086	0.122	0.225	0.225	0.222	0.230

Table 4.6: Number of operations as fraction of the maximum amount of work, Averaged for all datasets

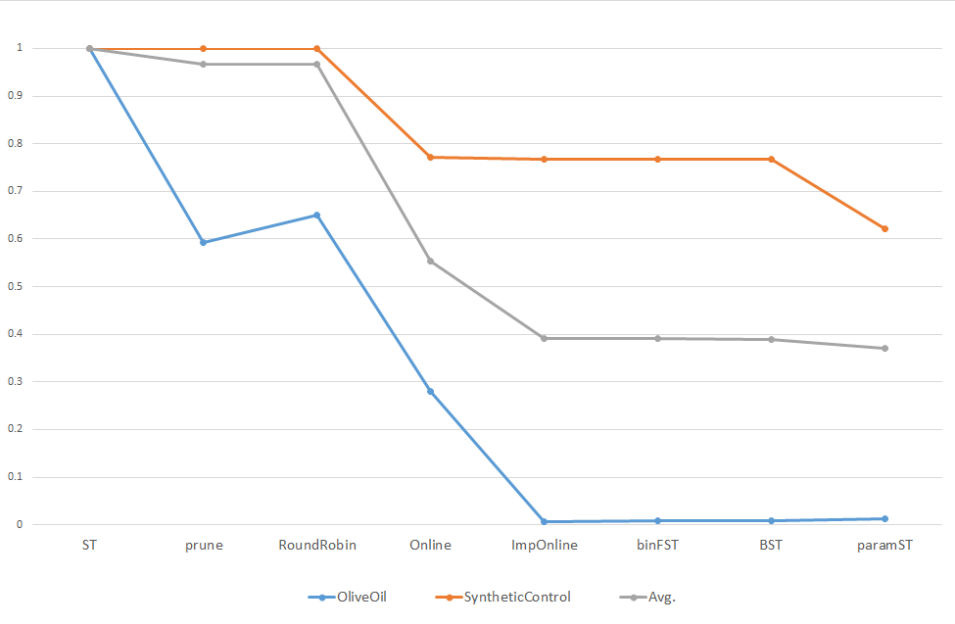


Figure 4.6: The Average total opCounts performed for the 7 different shapelets improvements. Average amount of work reduced, shown with the best and worst dataset. (Oliveoil,SyntheticControl)

4.9 Shapelet Distribution

The distribution of the shapelets in the final set is important in understanding what types of features the shapelet transform is finding and how the algorithm can be tuned. For each series the number of discrete values for shapelet length is dependent on the series length. To make the histograms more comparable, the shapelet lengths are normalized by the series length, and the counts are discretised into bins based on percentage of total series length. We selected 100 bins to discretise the data into, where each bin represents 1% of the series length. This enables a more fair comparison of long series to short series. Datasets with many cases, also have a larger number of shapelets found. This is because we set k to the number of cases n. Given that larger datasets have more shapelets, we chose to represent the amount of shapelets for a particular bin as a proportion of the total shapelets for a given dataset.

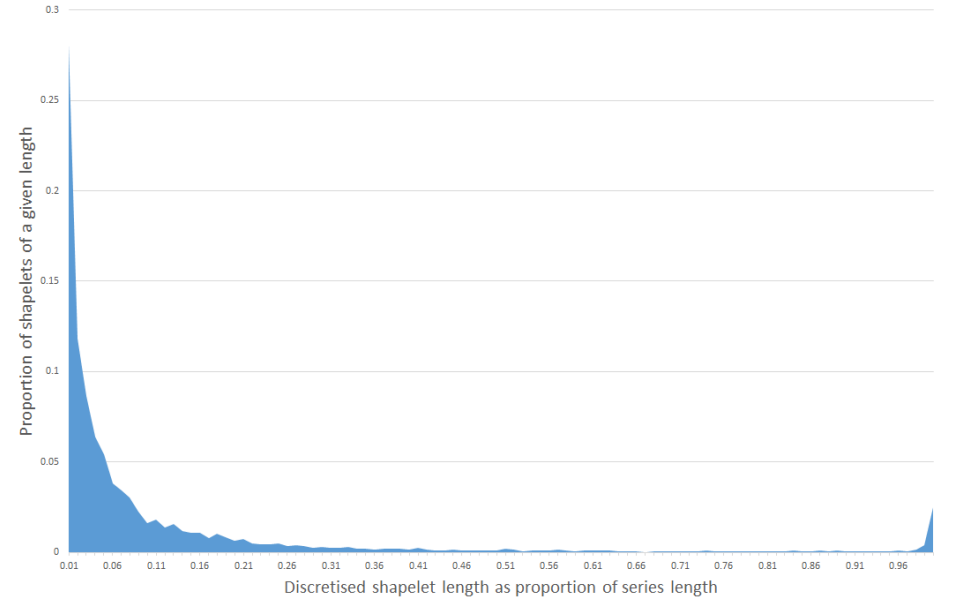


Figure 4.7: Normalised shapelet lengths with respect to series length for all shapelets in the set used in the transformation process

In Figure 4.7 we show the average distribution of the best shapelets found across all the datasets. The shapelets used to construct this histogram were constructed from our experiments in section 4.10. The distribution demonstrates that, against expectation shorter shapelets tend to be found and have higher quality than longer shapelets. It also highlights the fact that there is a small proportion, approximately 5%, of the shapelets in the final set that are the whole series. Lines et al. [70] described a heuristic approach to selecting minimum and maximum shapelet lengths, as computation power has increased. We opt to forgo this heuristic and consider the entire problem space from 3 to m. On closer inspection of the estimation function, the parameters selected often did not include shapelets that were less than 5% of the total series or shorter [46]. An example of this is with the UWaveGestureLibrary datasets.

dataset	minLength	maxLength
UWaveGestureLibraryX	113	263
UWaveGestureLibraryY	122	273
UWaveGestureLibraryZ	135	238

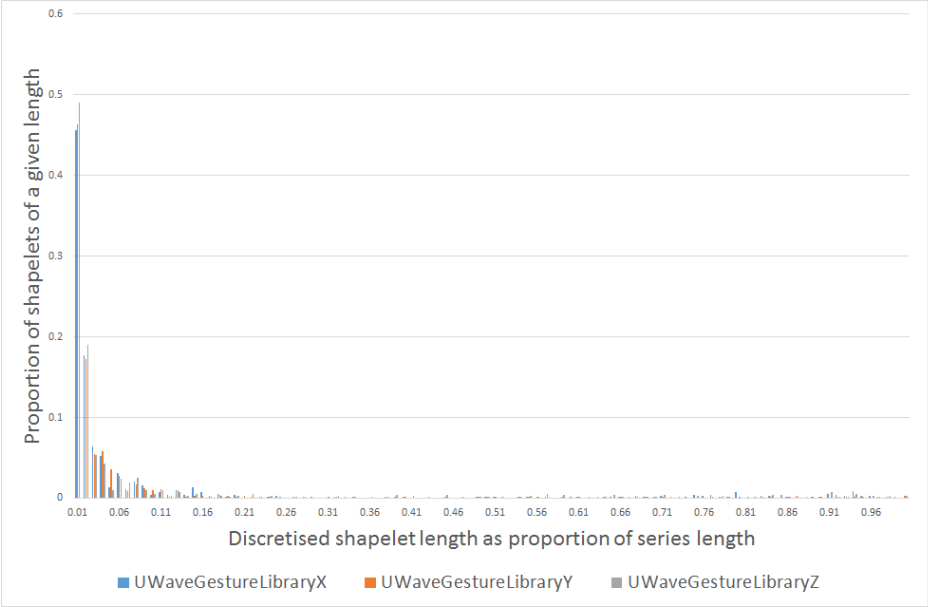


Figure 4.8: Normalised shapelet lengths with respect to series length for final shapelets for the datasets UWaveGestureLibraryX, UWaveGestureLibraryY and UWaveGestureLibraryZ

In Figure 4.8, the three datasets final shapelet counts are shown. Approximately 50% of the final shapelet set are found in the region that is less than 5% of the series total lengths. Considering the previous parameters and the selection algorithm it is reasonable to consider how only evaluating shapelets in the 35% to 85% range might produce a less representative transform.

4.10 Resampling Experiments

One of the major criticisms of recent time series classification work is that algorithms are often presented on select problems from the original UCR repository. The UCR-UEA repository was launched [23] as a larger standardized set of 85 datasets. These datasets are covered in more detail in chapter 3. As part of [8] and the launching of a larger shared problem set, the aim was to fully evaluate the best three shapelet methods from the literature.

The problem with only evaluating on 85 datasets however, and that which is typical of machine learning research in general, is that datasets tend to only have one default train and test split. These splits are often arbitrarily chosen when the dataset is designed/captured and released. This could have the consequence of making problems appear more difficult than they are, or may create biases on certain datasets for particular types of classifiers. It was proposed that creating 100 re-sampled problems for each dataset, where the train and test split are merged and stochastically sampled with respect to the original train/test distribution. This creates a large problem space to test on as each algorithm will be assessed on 8500 problems, whilst maintaining the number of cases for each class.

The aim of the second portion of this chapter is to fully evaluate ST_HESCA, Learn Shapelets and Fast Shapelets. Earlier we demonstrated there was no significant difference between the shapelet transform and learn shapelets when compared on the overlapping datasets published by Grabocka et al. [40]. These three algorithms are evaluated on 8500 problems, using the resampling technique outlined in chapter 3. These results contributed to the wider study presented in [8], where some of the published work produced by the other authors is used to evaluate the state of shapelet algorithms within the broader field of TSC.

One of the principles of this work was to contribute to an open source framework for time series classification [4]. This included the 85 datasets in a common format, in this case ARFF, source code for common algorithms and processing methods for re-sampling, filtering and processing time series data. Fast Shapelets and Learn Shapelets were both implemented in Java in conjunction with the WEKA frame work [43]. These methods both had input from the respective authors to ensure there correctness and in the case of Learn Shapelets some minor upgrades later proposed but not published at the time. To standardise the experiments for shapelets and across the wider work parameter setting was either performed by cross-validation where appropriate, and often with the original authors guidance, or parameters were fixed in accordance with the original work. In Table 4.7 we present the parameters for Fast Shapelets and Learn Shapelets.

	Parameters	CV Folds
LS	$\lambda \in \{0.01,0.1\}$, $L \in \{0.1,0.2\}$,$R \in \{2,3\}$	3
FS	$r = 10$, $k = 10$, $l = 16$, $\alpha = 4$	0

Table 4.7: Parameter Settings and ranges for Fast Shapelets and Learn Shapelets. Consistent with original authors parameters

For Fast Shapelets we mirrored the same parameters as those presented in [83]. For Learn Shapelets the parameters presented in [40] were varied dependent on the dataset. We opted to build three different parameter sets for λ, L and R which were chosen by pooling all parameters used in the original paper. On some of the largest datasets Learn Shapelets can take a long time to converge, and with a max run time of seven days on our HPC we opted to use three fold cross validation to ensure it completed successfully.

For the shapelet transform a small amount of search space reduction was required on the very large datasets, otherwise a full search would not have been feasible.

For some of the very large datasets two types of sampling were required. Firstly, a skipping mechanism with two stride parameters was used. These were defined for length skipping and position skipping. There is a large amount of redundancy in the shapelets evaluated in the transform. A shapelet of length 10, at position 0 and a shapelet of length 11 at position 1 have 9 values in common, although after z-normalisation the numerical values will be different. Our hypothesis is that with a small amount of skipping, the number of calculations can be reduced without significantly reducing accuracy. The most important factor to consider is that in previous experiments the min and max parameters were set through a shapelet length heuristic. This heuristic limited the range on the long datasets considerably. However, through our experiments we show that increasing the range of lengths available but considering fewer in the same region of the search space provides significantly better accuracy.

These stride parameters however are a heuristic and will not affect the worst case complexity of the algorithm. The other issue with the skipping

m	q
1500	32
1000	16
500	8
250	4

(a)

m	p
2000	8
1000	4
500	2

(b)

Table 4.8: Two tables for the skipping parameters. (a) contains length skipping, and (b) contains position skipping values

parameters is that as they become large this could negatively affect accuracy as too much of the search space is not evaluated. The modified shapelet transform that takes the skipping parameters p and q is described in algorithm 11. The criteria used for setting the skipping values are shown in Table 4.8. These values were chosen fairly arbitrarily, however, in chapter 5 we assess better ways to choose these and the findings are presented in [18].

Table 4.8 contains the cut off points for the different skipping values. For example if the series was of length 1200, its skipping parameters would be $q = 16, p = 4$, rounding down to the nearest length.

Algorithm 11 FindKBestShapeletsWithSkipping(**T**, min, max, k, p, q)

Where **T** is a set of Time Series.
$kShapelets = \emptyset$
for T_i **in T do**
  $seriesShapelets = \emptyset$
  for l **in** $\{min, ..., max\}$ **by** q **do**
    for pos in $\{0, ..., |T_i| - l + 1\}$ **by** p **do**
      $quality = checkCandidate(\mathbf{T}, T^l_{i,pos})$
      $seriesShapelets = seriesShapelets \cup \{T^l_{i,pos}, quality\}$
  $sort(seriesShapelets)$
  $removeSelfSimilar(seriesShapelets)$
  $kShapelets = merge(k, kShapelets, seriesShapelets)$
return kShapelets

In addition to the skipping parameters on some of the very large datasets sampling was performed to further reduce the shapelet search area. The

method for subsampling was aimed at reducing the number of time series, but without losing less represented classes. The smallest represented class is found. This class is sampled down to 25 series, the proportion of full size to sample size is applied to the rest of the dataset. If the sampling would reduce the dataset to less than 10% of the total size then the sampling is clamped at 10%. Any more than 10% seemed excessive, but this was arbitrarily chosen. In chapter 5 we further review sampling and its effects on the search space.

The mean accuracy over 100 folds for the three shapelet algorithms is presented in Table 1 (see appendix). At the bottom of the table we demonstrate the number of wins for the three algorithms and showing that on 71 out of the 85 datasets the ST_HESCA is better than LS and FS.

4.10.1 Results

The critical difference diagram for the results presented in Table 1 (see appendix) is presented in Figure 4.9. This critical difference diagram shows that the Shapelet Transform is significantly better than the other two algorithms. The critical difference for ST vs. the other best nine algorithms in time series classification is shown in Figure 4.10. The diagram shows that the shapelets method is significantly better than a large portion of the current state of the art and is only beaten by collective of transformation based ensembles which we presented in [6], of which the Shapelet Transform is an integral part.

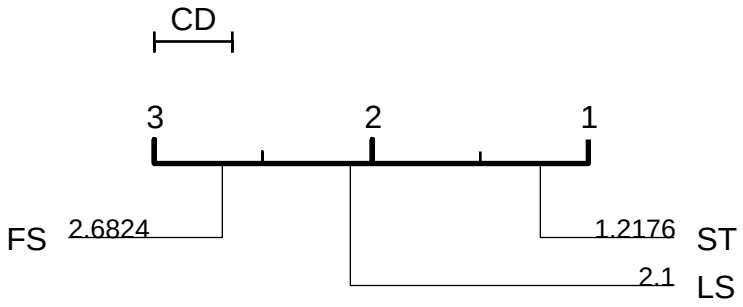


Figure 4.9: The critical difference diagram of Table 1, (ST is an abbreviation for ST_HESCA)

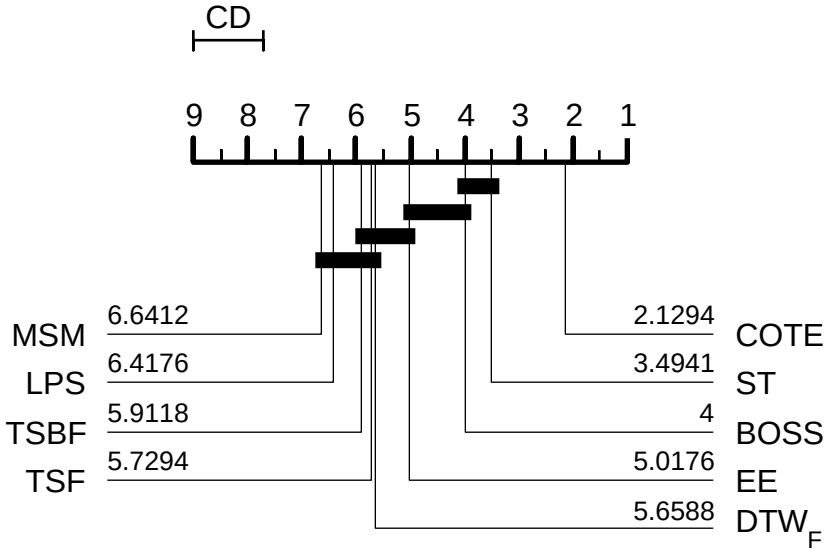


Figure 4.10: The critical difference diagram of the best 9 algorithms from [8]. These algorithms are described in section 2.2.

In a Wilcoxon signed rank test on fold 0 ST_HESCA was found to be significantly better than the original ST presented in [70]. In Table 2 (see appendix) we present the comparison of the results when comparing

between 73 datasets. In a Wilcoxon signed rank test, and the student t-test ST_HESCA is shown to be significantly better than the previous version of ST. These results are in line with our results presented earlier where we showed that ST was not significantly worse than Learn Shapelets based on the published data and in Figure 4.9 showed that with more datasets and using the improvements we proposed it was significantly better than Learn Shapelets. Over more datasets and using some small sampling rather than the length heuristic ST HESCA is significantly better.

COTEs accuracy has improved greatly since its original publication in [6]. In Figure 4.10 it is shown that COTE is the best algorithm for time series classification, and that both BOSS and ST HESCA are the joint second best algorithms. ST_HESCA is a core part of the COTE ensemble. The improvements proposed in chapter 4 can be attributed to some of the improvements in COTE. Unfortunately, we do not have the data required to quantitatively prove each individual ensemble components contributions, nor do we have the data to show how this changed from the original implementation to the current version. One of the main motivations for improving both the runtime and accuracy of the Shapelet Transform is that it should directly improve COTE. As the Shapelet Transform is one of the slowest algorithms present within the ensemble, reduction in the runtime requirements should result in COTE being more usable on larger problems.

4.11 Conclusion

In conclusion this chapter describes a Shapelet Transform that was better than the Shapelet Transform on multi class problems. We described a method for reducing the number of operations performed by the distance calculations that is used to calculate the quality of a shapelet.

We demonstrated that the balanced Shapelet Transform wins more often on problems with many class labels. We defined ST HESCA as a wrapper to simplify using the Shapelet Transform with the heterogeneous ensemble of classification algorithms, where the classifier can decide whether to use class balancing and binary shapelets or not.

We demonstrated that the changes made to the shapelet transform reduced the number of operations performed during a full search, and that on average there is no difference between a full search and the sometimes large cutoff values produced by the heuristic setting of min and max. However, closer inspection of the shapelets found showed they tended to be either less than 5% of the total series length or the series length itself (100%). In ST HESCA we will default the shapelet min and max parameters to be 3 and m confident that our speed ups in the $sDist$ function can offset the increase in the number of shapelets evaluated when compared to the heuristic length setting parameter, which we demonstrated can reduce accuracy.

In the second portion of this chapter (section 4.10) a new experimental methodology was proposed, where 8500 experiments for the Shapelet Transform, Fast Shapelets and Learn Shapelets were conducted on 85 time series problems. The aim of these experiments was to show which shapelet based method is the state-of-the-art and quantify how good all shapelet methods are when compared to other time series classification algorithms. In addition to this, these results provide an excellent point of comparison for new techniques to benchmark against. ST is compared with other results produced in the literature [8] where it is found to be the second best algorithm out of all reviewed. The best algorithm for time series classification was shown to be COTE, of which the shapelet transform forms an integral part of the ensemble.

The shapelet transform is compared with the original results presented in [70] and on the 75 datasets they have in common, the changes made to the shapelet transform have caused a significant improvement in classification accuracy. The aim of these experiments was to demonstrate that removing the shapelet length heuristic in favour of fixing the min and max to 3 to m. On datasets where the full enumeration is problematic we can use skipping instead. We demonstrated that some of the best shapelets were being missed in previous searches. However, because we have used a heuristic to find the shapelets, the global best shapelets may not have been found, and in some of the large datasets more improvements could be made.

There were some problems with the shapelet transform on the largest

of the TSC problems in the UCR-UEA archive, and some simple stride parameters were required to achieve reasonable runtime even on a HPC. With the success of shapelets in this study, this further motivates more research into runtime reductions. In chapter 5 the problem of quantifying shapelet runtime, and bounding searches to fixed time limits is explored. The aim is to give the shapelet search a fixed amount of time and produce similar or equal accuracy to the results presented in this chapter.

Chapter 5

Sampling the Shapelet Space

Contributing Publication

- A. Bostrom, A. Bagnall, and J. Lines. Evaluating improvements to the shapelet transform. *Knowledge Discovery and Data Mining, in Workshop on Mining and Learning from Time Series*, 2016

5.1 Introduction

Through a thorough and extensive analysis we have demonstrated that the Shapelet Transform is one of the best approaches to solving time series classification [8]. However, the brute-force search is not scalable for large or multivariate time series problems. The Shapelet Transform is $O(n^2m^4)$ and this leads to infeasible run time requirements for some of the datasets in the UCR-UEA repository (see chapter 3). In order to solve this problem we believe more algorithm development on the shapelet search is required. Alternative shapelet methods have either failed to provide accuracy that is not significantly worse when reducing run time requirements, or they find shapelets that are not present in the original data [8, 40, 83]. One of the major benefits of shapelet discovery, as opposed to learning shapelets, is the data-driven approach to finding class defining features and how they can be

mapped back to the original series for knowledge discovery. The problem with learning shapelets is there is no guarantee they exist in the data, and in the case of Fast Shapelets they represent many subsequences as a result of the aggregation. One type of problem Shapelets have had a large amount of success with is activity recognition. The concern in this problem area with derived shapelets, as opposed to found shapelets, is that the shapelet may not be constrained in the same way that the data capture is. This is especially apparent in human activity problems, where there is the potential to generate shapelets that describe impossible movements.

With these issues in mind in this chapter we described methods that find a set of shapelets that exist within the train data, and provide comparable accuracy with a full enumerative search, whilst requiring orders of magnitude less work to find. This will be accomplished by providing a contract approach to the shapelet transform. We define contract classification as a way of limiting the runtime of the shapelet transform such that when the limited runtime has expired a set of shapelets have been found, which may or may not be the global best solution. The goal of this work is to create search methods that find the best shapelets whilst being constrained to a fixed time limit. The deterministic nature of the shapelet transform means we are able to estimate the run time for a dataset.

In this chapter we will present four different search methods for use with a contract shapelet transform. Finding and evaluating shapelets is the constrained process in our contracting algorithm. To ensure comparable results the same classification method will be maintained (see chapter 4). The classification method we have used before is called the heterogeneous ensemble of simple classification algorithms (HESCA) (see section 4.5).

We define four shapelet search space techniques in section 5.4 where we show that we can find shapelets in two limited runtimes, searching for either one hour or one day. Our hypothesis is that we can maintain accuracy whilst reducing the amount of shapelets evaluated using adaptive searching.

This chapter is organised as follows. Section 5.2 presents a number of formulae for calculating worst-case run time complexity of the shapelet transform. This is based on previous work in chapter 4 and in [15, 18]. We

defined the operations in the Euclidean distance function as part of the shapelet evaluation process as the fundamental operation. Given a formal definition for calculating the approximate runtime of the shapelet algorithm, we then present the contracted shapelet transform algorithm which was initially presented in [18].

In the UCR-UEA archive there are 85 datasets available. In Table 5.1 and Table 5.2 we define two subsets which we identified as large and intractable problems dependent on either a one hour or one day runtime. Having defined the large datasets and a contract approach for the Shapelet Transform, the following sections describe four heuristic techniques to finding shapelets based on a fixed operation count.

The shapelet contracting is discussed in section 5.4 where we outline the four methods in subsections. In subsection 5.4.1 we describe a heuristic search for calculating a stride parameter for the sliding window function in the shapelet search. In subsection 5.4.2 we describe a heuristic search that randomly selects shapelets from the whole search space until the operation count limit is exceeded. In subsection 5.4.3 we describe a tailored Tabu search algorithm designed for shapelets, which blacklists areas of the search space. In subsection 5.4.4 we define the fourth heuristic search which uses a stochastic sampling method to constrain the search space as it iterates until the operation count is exceeded. Finally in section 5.5 we compare the four heuristic search methods, we perform pairwise analysis of the one hour and one day respectively.

5.2 Quantifying the time for enumeration

In this section the formula for calculating the number of shapelets and for calculating the number of fundamental operations in a dataset are presented.

Estimating the number of fundamental calculations for a dataset is a crucial component in estimating the run time on the large datasets. We define the runtime complexity function as the number of addition operations in the euclidean distance function when evaluating a single shapelet. In chapter 4 and in [15, 18] we demonstrated the effectiveness of the *opCount*

measure for comparing speed up techniques. To extend this work we wanted to define formulaes for calculating these values for any dataset. This will enable us to rank datasets by runtime, and search space size.

Considering the set of time series T. We calculate the number of shapelets using formula Equation 5.1

$$shapeletCount = \sum_{i=1}^{n} \sum_{l=1}^{m} (m - l + 1) \tag{5.1}$$

The summation from Equation 5.1 is expanded to Equation 5.2.

$$shapeletCount = \frac{nm^2 + nm}{2} \tag{5.2}$$

We define the total number of operations in a shapelet transform in Equation 5.3. To evaluate a single shapelet, we must evaluate it by comparing it to every other series, and slide it along that series calculating the euclidean distance.

$$opCount = \sum_{i=1}^{n} \sum_{l=1}^{m} (m - l + 1)^2 l(n - 1) \tag{5.3}$$

In Equation 5.4 we show the expanded summation from Equation 5.3 and in Equation 5.5 we describe a rearranged form for calculating the approximate number of cases based on a given *opCount*.

$$opCount = \frac{m(m+1)(m^2 + 3m + 2)(n-1)n}{12} \tag{5.4}$$

$$n \approx \sqrt{\frac{12opCount}{m(m+1)(m^2 + 3m + 2)}} \tag{5.5}$$

The final expanded formula in Equation 5.4 clearly demonstrates the worst case complexity of the shapelet algorithm as $O(n^2 m^4)$. In all of these equations we have included searching for shapelets from a length of 1 to m. In reality shapelets less than a length of three are inconsequential. For practical purposes we devised a set of formulas that calculated the *opCount* with respect to *min* and *max* parameters, as well as other metrics

for sampling which are covered in further sections. In Equation 5.6 the length and position stride parameters p and q have also been included in the equation. In Table 4.8 we presented an arbitrary heuristic for setting these values. Instead the aim is to define the runtime with respect to the datasets parameters and, from a given runtime requirement, derive the stride parameters to fulfill this contract. The min and max are assumed to be 3 and m: this is the case with all work. We denote the position skipping parameter as p and the length skipping parameter as q. We define the size of the set of possible lengths that exist with length skipping as $s = (m-3)/q$. The set of possible length values are:

$$L =< l_1, ..., l_s >$$

where

$$l_i = ((i-1)q) + 3$$

We define the opCount formula as:

$$opCount = \sum_{i=1}^{|L|} \left\lceil \frac{m - l_i + 1}{p} \right\rceil (m - l_i + 1)(l_i)(n-1)(n) \tag{5.6}$$

Which expands to

$$opCount = \frac{(m-3)(n^2-n)(m^3+7m^2-m(q^2-18q+27)+5q^2-24q+27)}{12pq} \tag{5.7}$$

We define one final equation for constrained runtime parameters for random shapelets. Given a fixed operation count $opCountTarget$ and the fixed $opCount$ we can derive the proportion of work required. We then calculate the amount of shapelets to be evaluated for random sampling methods, by multiplying the fixed shapelet amount (Equation 5.1) by this proportion.

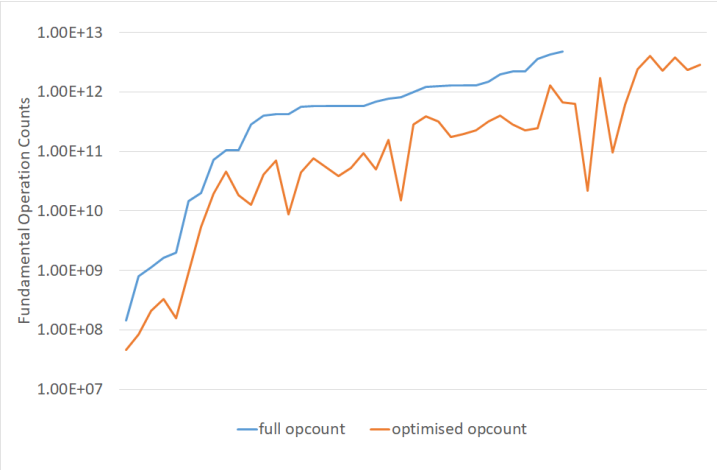
$$prop = \frac{opCountTarget}{opCount} \tag{5.8}$$

$$maxShapelets = prop * shapeletCount \tag{5.9}$$

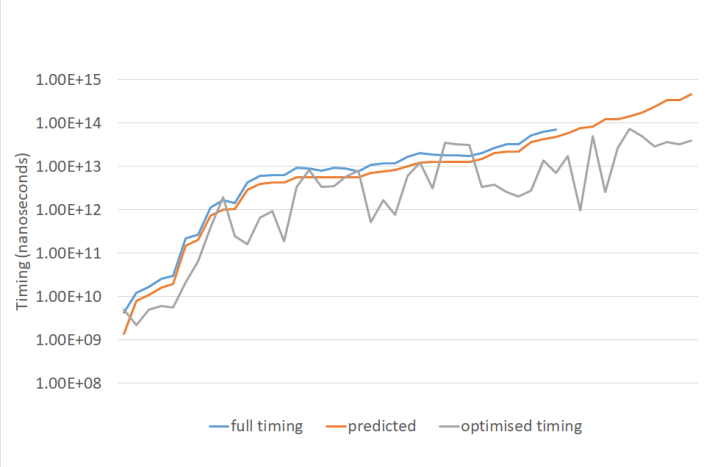
We presented two simple formulas for reducing the runtime of the shapelet search, these are by either having a fixed number of shapelets or by using stride parameters to avoid evaluating all shapelets. Having a fixed amount of shapelets will be useful when creating other heuristic search techniques as we will be able to anticipate the amount of the shapelet search space available.

All the time constrained experiments are either evaluated with a maximum train time of one hour, or one day. These train times reflected real-world expectations, and mirror other similar experiments in the literature [40, 83].

In Figure 5.1 we display the experimental operation counts matched alongside real-world timed recordings of the shapelet transform averaged over 10 runs. The aim was to ensure that we could realistically convert theoretical operation numbers to actual computer performance. The caveat is that this is dependent on a broad range of factors, some of which are out of our control. As is the case with all of these experiments we have tried to ensure they adhere to the one hour or one day runtime but they may not be these times exactly.



(a) opCounts



(b) Timing in nanoseconds

Figure 5.1: All datasets able to fully enumerate the shapelet set in one day runtime. We demonstrate the calculated opcounts and timing estimate against the recorded data on the full transform with no optimisations, and the full transform with current state-of-the-art optimizations.

The datasets for the one hour and one day experiments were carefully selected based on the size of the full shapelet set. We filtered these by calculating the amount of the shapelet space that could be explored in either

one day or one hour, and created a cut off point. Any dataset where 0.001% of the total shapelets cannot be calculated in less than the respective time limit are included. The cut off point is derived from the experiments we perform in section 5.3.

Two sub sets were created from the UCR-UEA archive, the one hour dataset contains 37 problems, and the one day dataset contains 20 problems. These are the largest datasets available in the archive. In Table 5.1 we present the one hour run time datasets. These problems are presented with the train and test instance sizes n, the length of the series m, and the number of classes C. In Table 5.2 we present the one day run time datasets, with the types of information.

datasets	n_TRAIN	n_TEST	m	C
CinCECGtorso	40	1380	1639	4
Computers	250	250	720	2
CricketX	390	390	300	12
CricketY	390	390	300	12
CricketZ	390	390	300	12
Earthquakes	322	139	512	2
ElectricDevices	8926	7711	96	7
FiftyWords	450	455	270	50
Fish	175	175	463	7
FordA	3601	1320	500	2
FordB	3636	810	500	2
HandOutlines	1000	370	2709	2
Haptics	155	308	1092	5
InlineSkate	100	550	1882	7
LargeKitchenAppliances	375	375	720	3
Lightning2	60	61	637	2
Mallat	55	2345	1024	8
NonInvasiveFetalECGThorax1	1800	1965	750	42
NonInvasiveFetalECGThorax2	1800	1965	750	42
OSULeaf	200	242	427	6
Phoneme	214	1896	1024	39
RefrigerationDevices	375	375	720	3
ScreenType	375	375	720	3
ShapesAll	600	600	512	60
SmallKitchenAppliances	375	375	720	3
StarlightCurves	1000	8236	1024	3
Strawberry	613	370	235	2
UWaveGestureLibraryAll	896	3582	945	8
UWaveGestureLibraryX	896	3582	315	8
UWaveGestureLibraryY	896	3582	315	8
UWaveGestureLibraryZ	896	3582	315	8
Wafer	1000	6164	152	2
Worms	181	77	900	5
WormsTwoClass	181	77	900	2
Yoga	300	3000	426	2

Table 5.1: One hour dataset list

datasets	n_TRAIN	n_TEST	m	C
CinCECGtorso	40	1380	1639	4
Computers	250	250	720	2
FordA	3601	1320	500	2
FordB	3636	810	500	2
HandOutlines	1000	370	2709	2
Haptics	155	308	1092	5
InlineSkate	100	550	1882	7
LargeKitchenAppliances	375	375	720	3
NonInvasiveFetalECGThorax1	1800	1965	750	42
NonInvasiveFetalECGThorax2	1800	1965	750	42
Phoneme	214	1896	1024	39
RefrigerationDevices	375	375	720	3
ScreenType	375	375	720	3
ShapesAll	600	600	512	60
SmallKitchenAppliances	375	375	720	3
StarlightCurves	1000	8236	1024	3
UWaveGestureLibraryAll	896	3582	945	8
Worms	181	77	900	5
WormsTwoClass	181	77	900	2

Table 5.2: One day dataset list

5.3 Sampling Shapelets

In this section, the aim is to demonstrate how sampling methods used to reduce the shapelet set size affect accuracy. Firstly, we define a simple sampling regime, which reduces the shapelet search space. Given that the amount of shapelets can be derived for a given dataset, this can be randomly reduced by sampling up to a fixed amount. Experiments were conducted on the UCR-UEA datasets over 10 folds. The parameters are a simple percentage reduction of the shapelet space. The simple formula $\frac{1}{10^p}$ where p is $2 \leq p \leq 7$ is used to determine sampling. On the datasets where p is either too small (fails to complete), or is too large (0 shapelets are considered). These results will be omitted. The aim is to show how accuracy changes as we consider less information.

The simplest approach for random shapelets is to uniformly randomly sample shapelets from the set of all shapelets. There are three parameters for selecting a random shapelet these are; series, length and position. In the case of randomly generating shapelet positions, this parameter is dependent on the length of the shapelet.

The random shapelet algorithm is described informally as; initially calculate the total number of shapelets available based on the datasets n and m values. The total shapelet count is sampled down to the given proportion parameter. Generate a set of random lengths, positions and series indexes from the set of all shapelets. This set of shapelets are all evaluated and the $kBest$ are maintained. These are then used to transform the original dataset.

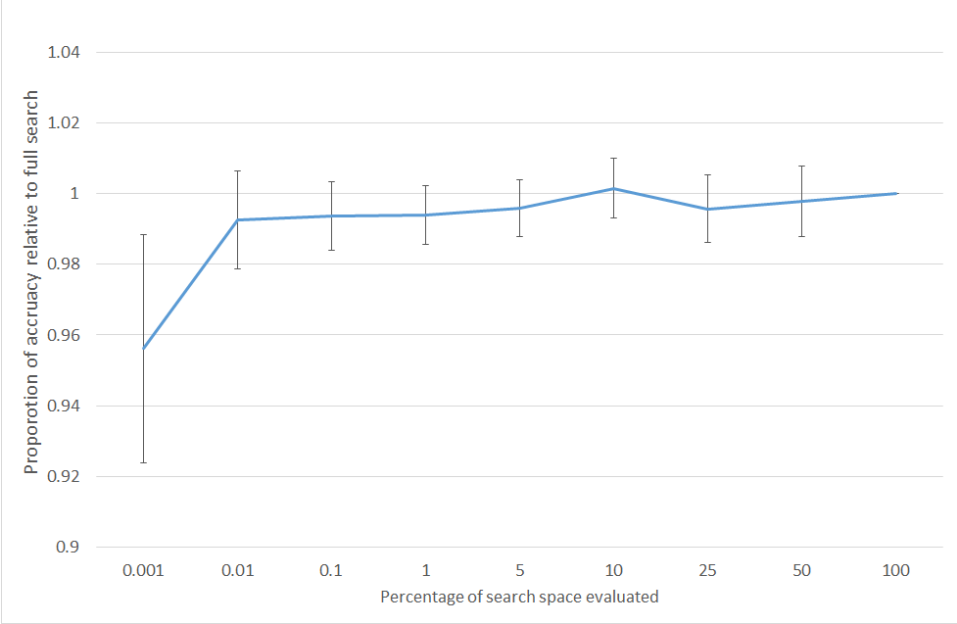


Figure 5.2: The proportion of accuracy relative to the full search. As the sampling on the shapelet search areas increase the accuracy becomes worse and the variance increases. This demonstrates how random sampling breaks down in the extreme case.

Figure 5.2 plots the average accuracy of the random shapelets for each

proportion proposed. This plot demonstrates that as we increase sampling, and reduce the amount of the shapelet search space we consider that accuracy is relatively unaffected. However, at some point, in the case of these experiments 0.01% of the shapelet search space the accuracy begins to break down and we get significantly worse results, that have a higher variance. The nature of the problem is that increasing the size of both series length and number of cases increases the amount of shapelets to search for and also increases the evaluation cost for each shapelet. So as problems increase in size, even looking at 0.01% of the search space in a reasonable time frame becomes infeasible.

5.4 Contract Sampling Algorithms for Shapelet Space

In this section we present four approaches to searching the shapelet search space. We define these search space algorithms as the skipping search, random search, tabu search and magnify search. We give motivations for each method and some of the problems they may have. As with all of these methods, some search methods may be more suited to particular types of data. Our aim is to find an approach that is on average better. However, the ability to tailor these algorithms to highly specific problems means that a tailored search could be better than the average case.

5.4.1 Skipping search

We initially conceived of a skipping approach to shapelet finding in [8] and in chapter 4 where the large problems were infeasible and we had to arbitrarily constrain problems to complete transforms. In previous work we had constrained the shapelet length to complete these problems in time. As we have shown in section 4.9 this method was suboptimal. In [18] we presented the initial results for the contract classifier evaluated on fold 0 of the UCR-UEA repository. The preliminary results for the one day run time were not significantly worse than the results presented in chapter 4 and [8].

Following on from these preliminary results we wanted to fully define the skipping search for contract classification.

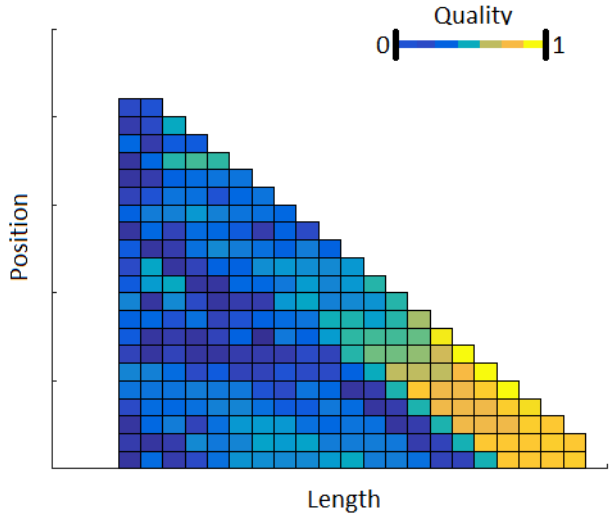
The skipping search is a simple method for finding shapelets. Given a runtime requirement we calculate how many operations the algorithm is allowed to make. From this given operation count we can derive how often we should skip along when performing the sliding window search of the shapelet space. This sliding window can skip on the length parameter or the position parameter. For example given a skipping parameter of two we would extract shapelets of lengths, 3,5,7 etc and positions 0,2,4 etc. For simplicity we keep the length and position parameters the same. This is because the number of solutions to our equation with two unknowns is often not unique, so we could have many permutations of length and position parameters to select from. Keeping the values the same ensures a unique solution and we do not think that either parameter is inherently better to minimise for better shapelet finding. We defined the skipping equation in Equation 5.6 and the algorithm was defined in chapter 4 in algorithm 11.

5.4.2 Random search

The second search algorithm we defined is the random shapelet search. One of the major downsides of the skipping search was that as problems became exceptionally large we could have stride parameters that skipped large chunks of the search space, potentially missing possible shapelets. Smaller shapelets were more likely to be missed because of the skewed distribution of shapelets based on length. For example the number of shapelets contained in a series given a set length is $m - l + 1$. This means that the distribution of the full set of shapelets are skewed towards smaller values of l. The reason skipping can be effective as a search method is because the best shapelets exist in neighbourhoods. However, when we are looking for small shapelets with large skipping parameters the likelihood that we will miss these neighbourhoods increases. In Figure 5.3 we show a quality map for the shapelets in the ItalyPowerDemand dataset. Where dark blue is low quality and light yellow is high quality. In this particular series the best shapelets tend to be close

to the whole length of the series.

Figure 5.3: A heatmap demonstrating the quality of shapelets found in a single series from ItalyPowerDemand



We illustrate another potential problem with skipping searches for shapelets with an an example. Given a series of length 100 in a single series there are 5050 possible shapelets $\left(\frac{m^2+m}{2}\right)$.

For length 3 there are 100-3+1 shapelets, which equals 98, for length 4 there are 97, and for length 5 there are 96. So with a skipping parameter of 2 skipping from 3 to 5 we avoid calculating 97 shapelets. This is approximately 2% of the shapelet space. For a length shapelet of 98 there 100-98+1, there are 3 shapelets. For length 99 there are 2 shapelets, and for length 100 there is 1. Skipping from length 98 to 100 means we skip 2 shapelets which is approximately 0.04% of the shapelet space. The distribution of the skipping values is unfairly biased towards evaluating long shapelets compared with short shapelets.

Random shapelets give equal weighting to all areas of the shapelet search space and so they should alleviate the major downside of a skipping search.

We informally describe the random shapelet search as:

- Given a fixed amount of shapelets to find, randomly generate the series, length and position parameters.
- Extract the generated shapelets and evaluate them.
- From these shapelets keep the k Best and use them to transform the dataset.

One of the problems with random sampling is that with a fixed number of searches, as the problem space grows large the search space becomes large and the distribution of the shapelets more sparse, therefore the likelihood of finding representative shapelets decreases. To validate this theory we conducted random searching experiments of fixed sizes (see section 5.3). In these experiments we saw that as the shapelet search space was reduced both the accuracy and variance of our results worsened. To counteract this problem we performed some subsampling experiments to explore how reducing the amount of samples that are considered, and concentrating the random search into a small area of the overall search space could reduce variance (see subsection 5.5.1).

5.4.3 Tabu search

To reduce the variance problems that a random search can have, we explore heuristic searching techniques that record areas of the search space and try to reduce the chance of evaluating shapelets that are similar. The Tabu algorithm is a heuristic search that was proposed in 1986 [37]. Tabu was designed as an algorithm which uses a local area search and both long and short term memory to avoid revisiting areas, or becoming stuck in local optima. Due to the nature of shapelets, we know that shapelets exist in neighbourhoods. In our particular implementation of Tabu we have large short term memory to find good localised shapelets with no global long term memory. Long term memory is not useful for shapelets because they are phase independent, therefore after each series is searched there is no

guarantee that shapelets of the same class will appear at the same position. We only check the previous best shapelets length and position across series, for the case where the similarity is not phase independent.

In algorithm 12 we present the pseudocode for Tabu search. Initially, the search is given a fixed number of shapelets per series based on the time constraints from the contract. The search starts by finding a random shapelet from the possible search space in the series. The neighbouring shapelets are then retrieved but if any of the neighbouring shapelets are in the *tabulist*, we abandon this local search area. If the random shapelet is in an unexplored region of the search space the surrounding area (neighbourhood) is evaluated and the local best shapelet is recorded. We then compare this local best shapelet to the best shapelets we have found so far and add it to the list of best shapelets. Finally, this shapelet is also added to the *tabulist* so the neighbourhood is not evaluated again. This process repeats until we have evaluated the allotted number of shapelets.

Algorithm 12 TabuSearch($\mathbf{T}$, T_i, min, max, $ShapeletsToEvaluate$)

Input: A set of time series $\mathbf{T}$, a series to search T_i, min, max and $bsfShapelet$
Output: A list of k Shapelets

```
1: shapelets ← ∅
2: tabuList ← ∅
3: shapeletsEvaluated = 0
4: currentShapelet = bsfShapelet
5: while ShapeletsToEvaluate > shapeletsEvaluated do
6:    currentShapelet = FindRandomShapelet(T_i)
7:    neighbouringShapelets = FindNeighbouring(currentShapelet, T_i)
8:    if tabuList.contains(neighbouringShapelets) then
9:       continue
10:   localBsfShapelet.Quality = EvaluateShapelet(currentShapelet)
11:   shapeletsEvaluated = shapeletsEvaluated + 1
12:   for all currentShapelet in neighbouringShapelets do
13:      currentShapelet.Quality = EvaluateShapelet(currentShapelet)
14:      shapeletsEvaluated = shapeletsEvaluated + 1
15:      if currentShapelet.Quality > localBsfShapelet.Quality then
16:         localBsfShapelet = currentShapelet
17:   if localBsfShapelet.Quality > bsfShapelet.Quality then
18:      bsfShapelet = localBsfShapelet
19:      shapelets ∪ bsfShapelet
20:   tabuList ∪ localBsfShapelet
21: return shapelets
```

5.4.4 Magnify Search

Having designed a shapelet specific version of the Tabu search, the aim was to evaluate another heuristic search that operates differently to Tabu. We propose magnify search as the fourth shapelet search algorithm. Magnify search constrains the random search space around the best shapelet, shrinking the search space as the algorithm iterates. Tabu search attempts to constrain the search space by reducing repeat evaluations, and exploits the property of shapelets existing in neighbourhoods by blacklisting based on proximity to previous evaluations. Magnify search performs a sparse stochastic sample of the search space it then reduces the search space around that region to

try and focus in on a particular area. With a depth parameter providing a way to evaluate large areas of space by increasingly shrinking the region of interest.

In algorithm 13 we describe the magnify search in pseudocode. The initial search space is considered at depth 0. A list of random shapelets is generated the size of which is defined by the *max* depth and the total shapelets set by the contract. All of these shapelets are evaluated and the best so far becomes the centroid. The search space is reduced by half around this shapelet. The method repeats until a max depth is reached, where by on the last stage these shapelets are recorded in the best so far list.

Algorithm 13 MagnifySearch($\mathbf{T}$, T_i, min, max, $ShapeletsToEvaluate$)

Input: A set of time series $\mathbf{T}$, a series to search T_i, min, max and $bsfShapelet$

Output: A list of k Shapelets

```
1: shapelets ← ∅
2: minL = min
3: maxL = max
4: minP = 0
5: maxP = max − min + 1
6: lengthWidth = (maxLength − minLength)/2
7: posWidth = (maxPos − minPos)/2
8: for depth ∈ {1, ..., MaxDepth} do
9:    bsf_Shapelet
10:   shapeletsEvaluated = 0
11:   while ShapeletsToEvaluate > shapeletsEvaluated do
12:      shapelet = FindRandomShapelet(T_i, minL, maxL, minP, maxP)
13:      shapelet.Quality = EvaluateShapelet(shapelet)
14:      shapeletsEvaluated = shapeletsEvaluated + 1
15:      if shapelet.Quality > bsfShapelet.Quality then
16:         bsfShapelet = currentShapelet
17:      if depth == MaxDepth then
18:         shapelets ∪ shapelet
19:   lengthW = lengthW/2
20:   posW = posW/2
21:   minL = |bsfShapelet| − lengthW
22:   maxL = |bsfShapelet| + lengthW
23:   minP = bsfShapelet.startPos − posW
24:   maxP = bsfShapelet.startPos + posW
25: return shapelets
```

5.5 Experimental Comparison

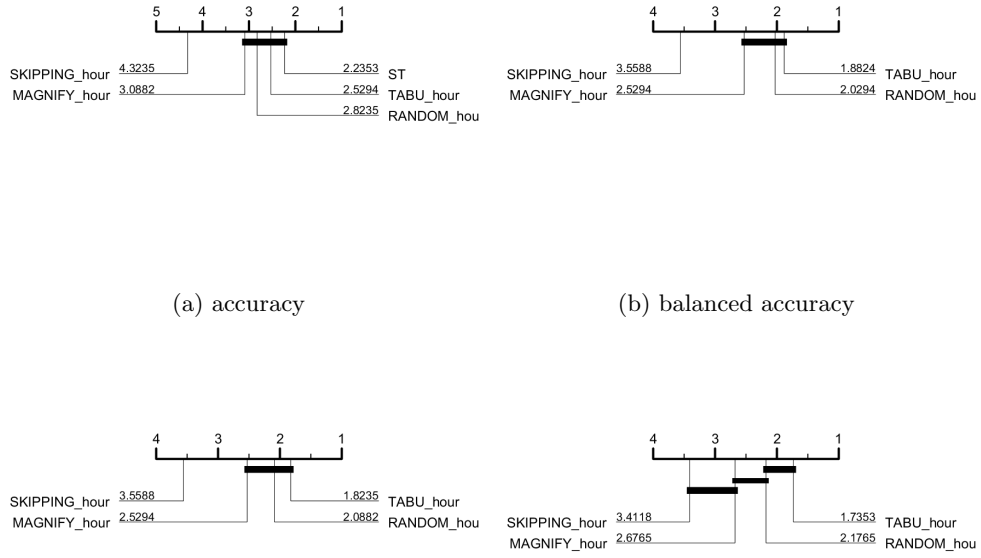
The main goal is to find robust heuristic methods for finding shapelets which are not significantly worse than the full search but are significantly faster. In this section we evaluate the four searches on both the hour and day datasets presented in Table 5.1 and in Table 5.2.

The experimental setup is as follows, initially four searches are compared

to the current state of the art ST results. These ST results are from the experiments performed in chapter 4 and are reported in [8].

Thirty sets of evaluations were performed for each dataset, capturing the accuracy and calculating the variance across multiple runs. As we have recorded the predictions for these calculations we also calculate additional statistics for evaluating the searches. In particular we calculate the balanced accuracy and the f-score.

Figure 5.4 presents the critical difference diagram for the four searches using a one hour run time, measured against ST. For the one hour evaluation it is shown that there is no significant difference between Tabu, Magnify or Random and our baseline ST (full).



(a) accuracy

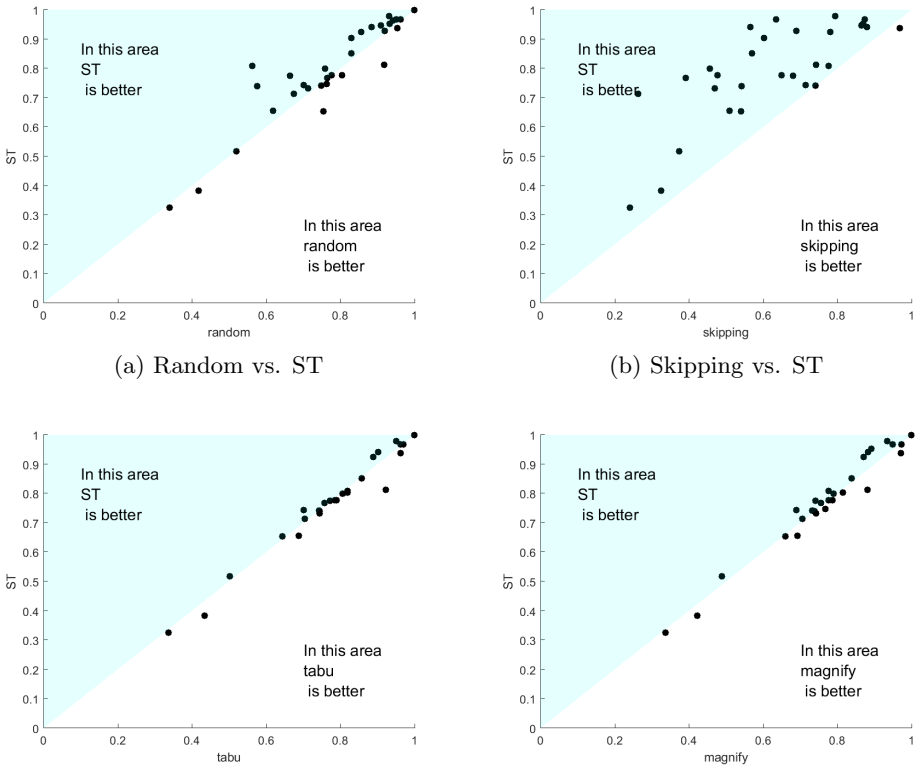
(b) balanced accuracy

(c) f score

(d) AUROC

Figure 5.4: A critical difference diagram comparing the four search algorithms, with a runtime of one hour, and the Shapelet Transform via error. Three additional critical difference diagrams compare the four search algorithms by, balanced accuracy, f score and AUROC.

Figure 5.5 presents the pairwise scatter plots of the four search algorithms compared to ST. These plots highlight the fluctuation within the random search when compared to ST. For both magnify and tabu search the accuracy results have little variance and are tightly aligned along the diagonal, this indicates little difference between the reported results. With the Random results in some cases there are large differences between ST and the random search. In some cases this greatly benefits the classification accuracy and in others produces worse results.



(a) Random vs. ST

(b) Skipping vs. ST

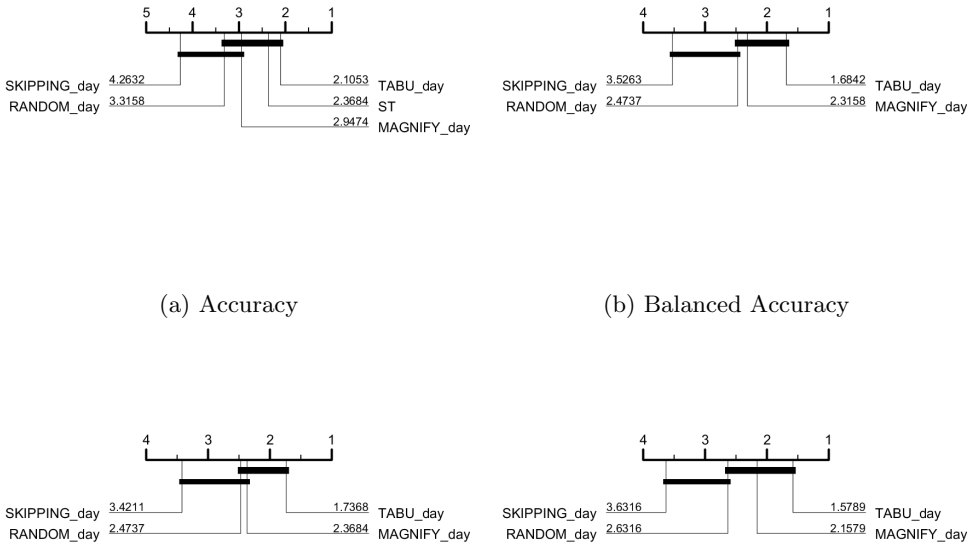
(c) Tabu vs. ST

(d) Magnify vs. ST

Figure 5.5: A set of four pairwise scatter plots demonstrating the accuracy of the respective search algorithms with a runtime of one hour compared with the Shapelet Transform

Figure 5.6 presents the one day runtimes of the same four searches and the ST results. In this critical difference diagram the problems discussed earlier with skipping search become evident. On the one hour run time ST was significantly better than skipping, but with more time the searching is less brittle and is able to perform as well as the full search, which was shown in [18]. It is worth noting that the tabu search has the highest rank in both cases but is not significantly better than the other heuristic searches. As the search is given more time, it is able to form a larger *tabulist* and avoid poor

search areas.



(a) Accuracy

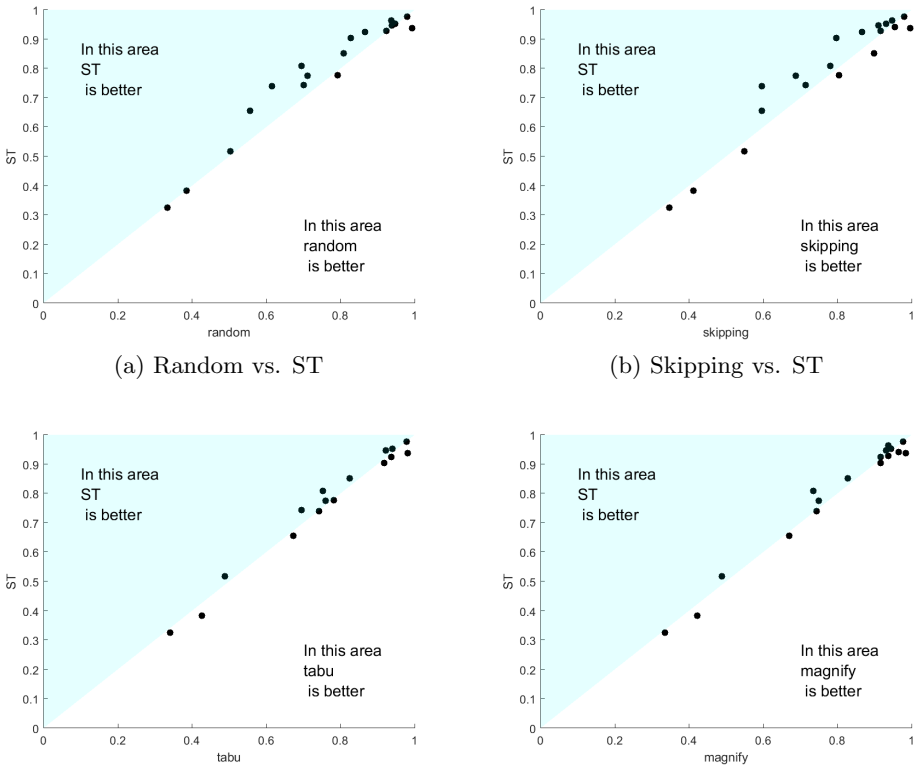
(b) Balanced Accuracy

(c) F Score

(d) AUROC

Figure 5.6: A critical difference diagram comparing the four search algorithms, with a runtime of one day, and the Shapelet Transform via error. Three additional critical difference diagrams compare the four search algorithms by, balanced accuracy, f score and AUROC.

Figure 5.7 presents the pairwise scatter plots of the four search algorithms compared to ST. Whilst random search is not significantly worse than tabu, magnify or ST its overall rank is lower, and this indicates some of the problems highly random searches can have. This also motivates why more specialised heuristic searches such as tabu and magnify are required.



(a) Random vs. ST

(b) Skipping vs. ST

(c) Tabu vs. ST

(d) Magnify vs. ST

Figure 5.7: A set of four pairwise scatter plots demonstrating the accuracy of the respective search algorithms with a runtime of one day compared with the Shapelet Transform

5.5.1 Subsampling Random Shapelet search

To alleviate some of the problems that can occur with random sampling we explored subsampling the series before searching for shapelets to concentrate the evaluation on a smaller area. In Figure 5.8 we present the two critical difference diagrams for our experiments. We performed 30 fold evaluations with three levels of subsampling where the shapelets available come from either 10%, 25% or 50% of the series and are evaluated on that smaller set too. Our results show that there was no significant reduction in accuracy

for any of the random methods when compared to ST. Subsampling is an effective way to reduce the size of n without affecting accuracy, and suggests that some of the sampling in chapter 4 should not have negatively affected the accuracy of the classifiers.

(a) One Hour Random Subsampling (b) One day Random Subsampling

Figure 5.8: A pair of critical difference diagrams presenting the preliminary results of comparing 3 types of random subsampling with ST

5.6 Case Study: HeartbeatBIDMC

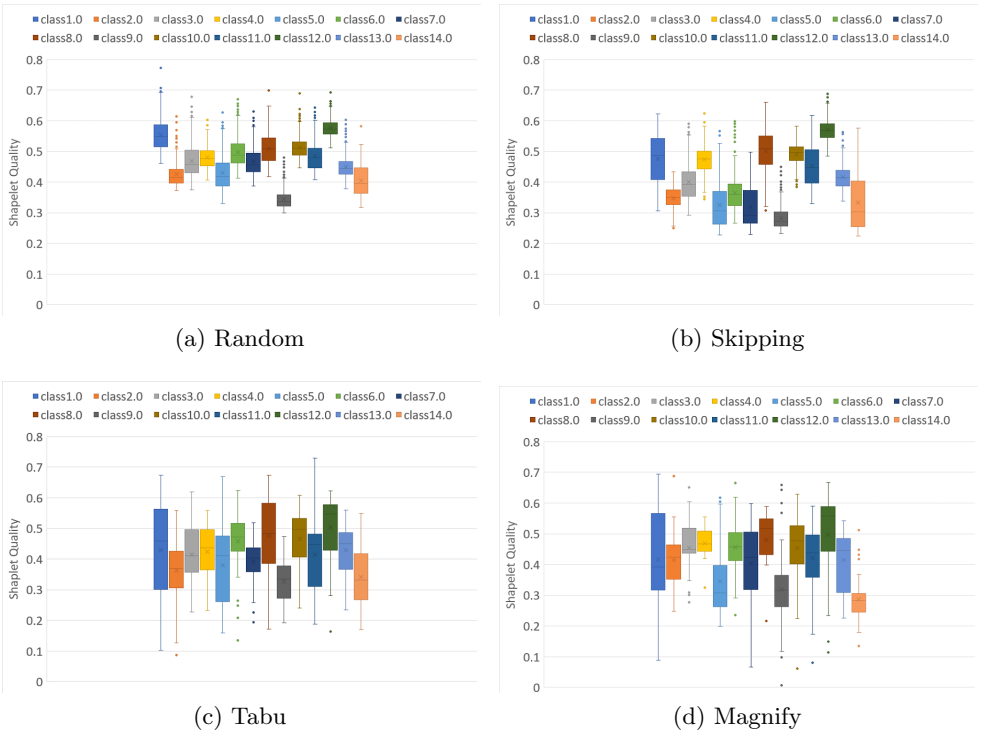
The dataset heartbeatBIDMC is the longest dataset available in the UCR-UEA archive. It was first presented as a time series classification dataset in [94]. HeartbeatBIDMC is a set of 600 time series that are 3750 values long, consisting of 14 patients who suffer from congestive heart failures. The recordings are off ECG and contain high levels of noise, large variance even within the same class. As this is one of the largest datasets available in the time series classification community, it presents an ideal opportunity to demonstrate the sampling techniques presented in this chapter, and to consider the sets of shapelets found and how they effect accuracy. This dataset was first evaluated using the skipping mechanism and the results were present in a small case study in [18] where we considered skipping shapelets to solve the problem in a one day runtime.

With the four search algorithms presented in this chapter we compare the results of those experiments on 10 folds with a one day runtime. The accuracy of the methods are presented in Table 5.3.

datasets	SKIPPING	RANDOM	MAGNIFY	TABU
HeartbeatBIDMC	0.955 (0.01)	0.99 (0.01)	0.98 (0.01)	0.974 (0.01)

Table 5.3: Table of average Accuracy conducted over 10 folds along with the standard deviation

In Figure 5.9 four box and whiskers plots of the quality of the best shapelets found for each of the fourteen classes on the heartbeatBIDMC dataset are presented. The aim of these diagrams is to demonstrate the correlation between finding better quality shapelets and with improved classification accuracy. In Table 5.3 there are four accuracies, one for each of the searches performed which are averaged over 10 folds. The skipping search having the worst accuracy and the random search having the best. Considering each of the plots for each search it is clear that the random search has more high quality shapelets than those found in the skipping search and the range of shapelets found is more concentrated, but with a large number of outliers. The skipping search has fewer high quality shapelets and more low quality shapelets across almost all of the classes. The range of shapelets for the tabu and magnify search are slightly more spread however the quality of shapelets found by these searches tends to contain less outliers, which is expected because of the area type searches they perform. The problem with the tabu and magnify search on this type of dataset is that, because the series is very long, and has a large number of cases concentrating on a few series means that the shapelet transform does not see much of the total dataset. With the random search because the pool of shapelets is unrestricted there is a lot of variance in the quality, but if the search can find a small amount of high quality shapelets this can improve accuracy.



(a) Random (b) Skipping

(c) Tabu (d) Magnify

Figure 5.9: A set of four box and whiskers plots showing the quality of shapelets collected for each of the fourteen classes in the heartbeatBIDMC dataset.

5.7 Conclusion

In this chapter we discussed and presented the idea of a contract classifier, where we can bound the runtime of our shapelet search by deriving the number of total operations and sampling down to a fixed runtime. We then gave an overview of four heuristic search techniques that have been considered one of which we presented in [18]. We evaluated these four search techniques on datasets we identified through our preliminary experiments as being particularly large and problematic for full search. We produced two sets of problems for one hour and one day evaluations using our proposed

contract classification framework.

We initially presented the results comparing one hour and found that tabu had the highest rank overall and that the skipping search was particularly brittle when the contract times resulted in large skipping parameters. Evaluating on the one day datasets showed that the skipping search was not significantly worse than of the other techniques, however we believe it is the weakest of the four. Following on from these we saw that the two meta-heuristic searches we designed and created were able to perform very well. The random search with or without sub sampling appears to be the most flexible approach with little issue maintaining accuracy against ST even on the one hour run times. The meta-search heuristics can have issues on very large problems where n and m are particularly large and there are few shapelets to analyse per series. In these cases, sampling series becomes necessary to concentrate the searches. In these particular cases randomly selecting shapelets should be considered. As we have shown the random search to be particularly effective and does not require parameters to use, we will opt to use random searching when performing contract classification in future work. Only using tabu when variance becomes problematic and we can effectively sample.

Chapter 6

Multivariate Shapelet Transforms

Contributing Publication

- A. Bostrom and A. Bagnall. A Shapelet Transform for Multivariate Time Series Classification. *ArXiv e-prints*, 2017

6.1 Introduction

Multivariate time series classification (MTSC) has gained traction in recent years, although the majority of work in time series classification has focused on the univariate case. For univariate TSC a class label is assigned to a single series, in MTSC each class label is assigned multiple series. It is commonly claimed that transitioning to multivariate from univariate is trivial (e.g. [99]). However, we do not believe this is necessarily true.

For the datasets we consider in this thesis, each case has a class label that is a single value which does not change over time.

We formally define multivariate time series classification dataset as $\mathbf{MT}=\{MT_1, MT_2, ..., MT_n\}$ which is a set of n multivariate time series. A single multivariate time series $MT_i = \{\{T_{i,1}, T_{i,2}, ..., T_{i,d}\}, c\}$ is a set of d univariate

time series with a single class label. Each series in a multivariate instance is described as $T_{i,j} =< t_{i,j,1}, t_{i,j,2}, ..., t_{i,j,m} >$ where we define the length as m. For simplicity of notation we assume all series in the dataset are the same length.

Multivariate time series classification has many practical applications. These can range from medical problems, such as electroencephalogram (EEG), finance, multimedia, human activity recognition (HAR) and gesture recognition. In Figure 6.1 we demonstrate a simple representation of the X,Y and Z data for two series in the UWaveGesture problem, the first being from class 1, and the second from class 8 [73].

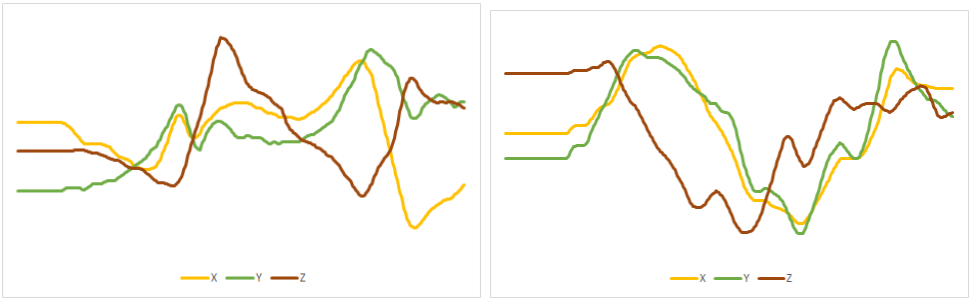


Figure 6.1: Examples of Class 1 and Class 8 with their respective X, Y and Z multivariate series from the UWaveGesture dataset

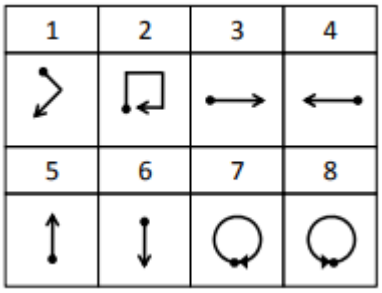


Figure 6.2: Class Labels for the UWaveGesture dataset. Image taken from [73].

This chapter describes new approaches to multivariate time series classification. The constraints of this work were that time complexity of shapelets must not exceed simple multivariate methods techniques that are used for extending univariate algorithms. The methods we propose all scale linearly with respect to the number of dimensions. All of the multivariate methods we propose were required to build on top of the work produce in chapter 4 and in chapter 5. In chapter 2 we reviewed the state of multivariate research and some existing techniques. section 6.2 describes the initial experiments on the multivariate datasets and how we can gather benchmarking data on these problems. In chapter 2, section 2.17 the multivariate dynamic time warping techniques used in this chapter are described. In section 6.3 some of the problems with scaling the shapelet transform using simpler techniques are discussed and the need for specific multivariate variants are explained. We present three multivariate shapelet algorithms in section 6.4 and section 6.5. Finally in section 6.6 we conduct an experimental comparison of all the algorithms we have discussed, followed by our conclusions in section 6.8.

6.2 Benchmark Experiments

Before creating and testing new shapelet approaches to MTSC we wanted to establish the state of current MTSC research and construct a unified problem space within the UCR-UEA repository specifically for multivariate problems. In [8] and in chapter 4 an experimental process was designed for comparing univariate classifiers on a unified problem space across 100 resamples of each dataset. To ensure a high quality analysis of shapelet algorithms, and of future problems in this field we first need to establish a benchmark set of results with which to compare too. Based on the experimental evidence for the univariate experiments, and the results presented in [65] we chose to benchmark on the five simple classifiers that form the core of the heterogeneous ensemble of simple classification algorithms (HESCA) (see section 4.5). We chose to use the five constituents of HESCA for three reasons. Firstly, HESCA is the main classification model that is paired with the Shapelet Transform. Secondly, the constituents chosen for HESCA were selected through a rigorous evaluation

of many different classifiers [65]. Finally, each of these algorithms detects different types of features, and so could reveal the underlying structure within the data. The five algorithms chosen are: a support vector machine with a polynomial(quadratic) kernel [81]; a one nearest-neighbour with dynamic time warping[85, 91]; a multi-layer perceptron[90]; a random forest [19]; and rotation forest[89].

These initial experiments will ascertain the difficulty of the MTSC data. By training a set of simple classifiers on the multivariate datasets, we can provide a benchmark to compare to the current state of the art, as well as, evaluating the overall performance of new shapelet methods.

We use a simple independent dimension ensemble which, will train a separate classifier on each dimension. We opted to keep the ensemble as simple as possible, forgoing any form of weighting of predictions via cross-validation to ensure a quick train time and easy to reproduce benchmark. In addition to an ensemble benchmark we also concatenate the dimensions into a univariate series and then train and build a single classifier.

On both concatenation and ensemble models, we perform no parameter tuning on any of these algorithms and set the algorithms parameters to typical defaults. For univariate problems parameter tuning a single model on a univariate dataset is a relatively simple task. For the ensemble case, however, tuning parameters for each model on each dimension can be a very time consuming process, and is counter to the purpose of simple and easily reproducible benchmarking.

The experiments are conducted on 24 datasets which were introduced in chapter 3. These have either been selected from within the literature, or constructed from data gathered at UEA. The multivariate TSC archive is available from the UCR-UEA repository [23]. The datasets are converted into the WEKA arff format [43] and all code and experiments are reported and stored on [5, 4].

6.3 Scaling the Shapelet Transform for Multivariate data

In chapter 4 and chapter 5, we described the shapelet transform for univariate TSC including new heuristic improvements to the distance calculations and early-abandon. A contract shapelet search was described which samples the space of possible shapelets. The aim was to mitigate the shapelet algorithm's prohibitive runtime complexity of $O(n^2m^4)$.

With multivariate time series classification, the runtime complexity problem gets worse. A naïve concatenation of the multivariate data into univariate series increases each series length to dm, assuming equal length dimensions. Consequently, the current Shapelet Transform on concatenated multivariate data has a runtime complexity of $O(n^2(dm)^4)$. For long or high dimensional data, this is clearly not scalable.

Many of the multivariate datasets presented in chapter 3 are infeasible to fully enumerate, even with the new methods we presented in chapter 4. Furthermore, for some of the very large datasets the number of shapelets it is possible to evaluate with a contract approach, is a tiny fraction of the full space. The dataset PEMS contains 267 series, which have 144 dimensions where each dimension is 964 values long, a full enumeration on the concatenated series would require approximately $2.6 * 10^{25}$ operations, which is estimated at 10^11 years.

As well as their predictive power, one of the main benefits of using shapelets is the interpretability they provide. The shapelets found in concatenated data are not interpretable because they are dependent on the ordering of the concatenation process. The often arbitrary ordering of the concatenation can completely change the types of shapelets found, and their ability to separate the data into their respective classes. As the number of dimensions increases this problem is exacerbated.

Ensembling the Shapelet Transform appears to be a simple solution to these problems with multivariate data. Keeping the dimensions separate avoids increasing further the already worst-scaling factor of the transform, m,

and restricting the shapelets to be within individual dimensions maintains their intuitive interpretability. However, on very large problems, where sampling is required, this approach is more difficult. Initially we considered training individual Shapelet Transforms on each dimension, where the number of shapelets that can be evaluated in the time frame is evenly split between each dimension. This method is bounded by $O(n^2m^4d)$. However, on some of the very large datasets, especially the highly dimensional PEMS dataset, the number of shapelets available to each transform is still very low. Performing any form of cross-validation to weight the importance of dimensions is infeasible and we cannot benefit from techniques such as bagging of boosting to improve overall ensemble performance. The way in which contract classifiers handle cross-validation is open to interpretation. If the contract calculations required cross-validation to be taken into account, certain time frames become impossible on some datasets. Other considerations for contract ensembling could be giving different dimensions variable amounts of time, depending on the importance of the dimension. We believe multivariate contracted ensembling is a large open ended research question, and worth significant exploration, but is out of the scope of this piece of research.

6.4 Independent Shapelets

The first multivariate shapelet method is called Independent Shapelets (ST_IND). This algorithm finds single dimension shapelets from any dimension. It then assesses the shapelets quality against the other series via sliding the shapelet along the same dimension in the multivariate series. Once the k best shapelets have been found, they are used to transform the original dataset. Using the same distance method, we can transform the multivariate dataset in a k by n matrix, where we find the respective distance of the shapelets to each series. The runtime complexity of this algorithm is $O(n^2m^2d)$.

The motivation for this method is that in some multivariate datasets the class defining feature may occur in only one dimension, and it could even be

independent of dimension. The shape of the feature is the class identifier not its position or dimension. This method should extract identical shapelets that would occur in an ensemble version of the Shapelet Transform, but will build a single transform rather than multiple transforms.

This method is most suited if you have multiple dimensions from different types of data recording where the dimensions are unrelated. One of the datasets we present MVMotionAG contains three dimensions of accelerometer data, and three dimensions of rotational(Gimbal) data. In some of the activity recognition the rotational data is completely independent of the movement information.

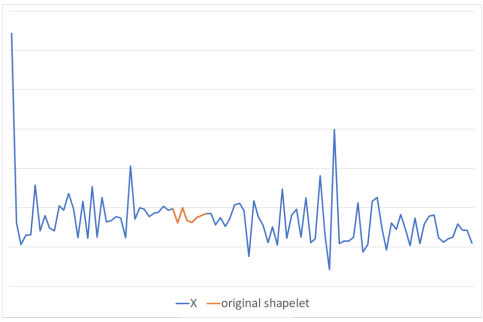
In algorithm 14 we formally define the full search for Independent Shapelets. We loop through each instance MT_i in the dataset **MT**, for each data series in MT_i, the algorithm loops over the series considering all lengths between min and max, and all positions between 0 and the data series minus the current length plus 1. The subsequence is extracted in the variable *shapelet*. This is then compared to all other series using $checkCandidate$, a this is shown in 6.3a. In algorithm 15 we calculate the information gain for the shapelet passed in. For each series in the dataset the shapelet is compared to the same dimension that the shapelet is extracted from. This shapelet is slid along the series in $sDist$ finding the minimum distance to match with, which is demonstrated in 6.3b. The shapelet is compared with individual normalised subsequences in the series, which we demonstrate in 6.3c. $sDist$ calculates a single distance value for a shapelet when compared to a single multivariate instance and the set of distance values for each series is used to construct an orderline and the calculate shapelets information gain which we have described in greater detail in chapter 4.

Algorithm 14 FindBestIndependentShapelets($\mathbf{MT}, min, max$)

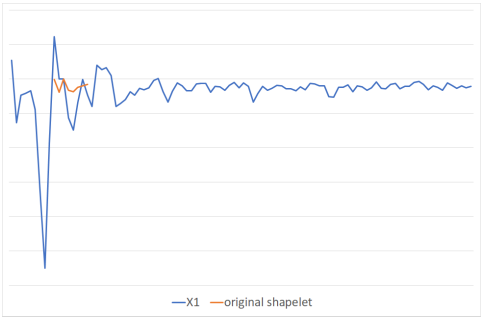
Where **MT** is a set of Multivariate Time Series.
$KShapelets = \emptyset$
for all MT_i in **MT do**
  $seriesShapelets = \emptyset$
  for $j = 1$ **to** d **do**
    for $l = min$ **to** $\mathbf{MT}_{i,j}$ **do**
      for $p = 1$ **to** $max - l + 1$ **do**
        $shapelet = \mathbf{MT}^{l}_{i,j,p}$
        $quality = checkCandidate(\mathbf{MT}, shapelet, j)$
        $seriesShapelets = seriesShapelets \cup \{\mathbf{MT}^{l}_{i,p}, quality\}$
  $sort(seriesShapelets)$
  $removeSelfSimilar(seriesShapelets)$
  $kShapelets = merge(k, kShapelets, seriesShapelets)$

Algorithm 15 checkCandidate($\mathbf{MT}, shapelet, d$)

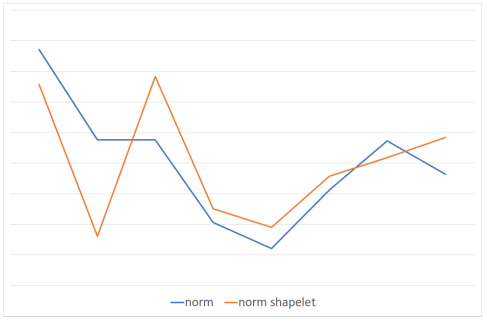
Where **MT** is a set of Multivariate Time Series.
$dist$
Where O is an orderline.
for MT_i in **MT do**
  $dist = sDist(shapelet, \mathbf{MT}_{i,d})$
  $O \cup dist$
return informationGain(O)



(a) Shapelet extraction



(b) Shapelet matching



(c) Normalised distance

Figure 6.3: An example of extracting a single shapelet from a many dimensional series, and comparing it to a different series of the same dimension

.

6.5 Finding Multidimensional Shapelets

ST_IND extracts single shapelets from a single dimension. We now consider the case of extracting shapelets that span across all dimensions. This is demonstrated in both 6.4a and in 6.5a. The main difference between Multivariate shapelet extraction and univariate extraction is we extract a shapelet from each dimension when given a length and position. For ease of mathematical notation we assume all dimensions are the same length.

For the multidimensional search we extract subsequences from within a multivariate time series as a block that spans all dimensions. These

sequences are are compared with the other series in the dataset in the function *checkCandidate*. The type of *sDist* function in the Shapelet Transform depends on the methods we present in subsection 6.5.1 and in subsection 6.5.2.

6.5.1 Multidimensional Dependent Shapelets

The first multivariate shapelet method is called multidimensional dependent shapelets (MST_D). This method extracts multi-dimensional shapelets, that are then compared to the other multivariate series, maintaining the phase across channels.

In algorithm 16 we describe the process of calculating the distance when comparing a multivariate series to a single multivariate series. The multivariate shapelet is slid along the time series from position 0 to position $m - l + 1$, this is shown in 6.4b. 6.4b also illustrates how the shapelet is measured across the dimensions as a phase inter-dependent band.

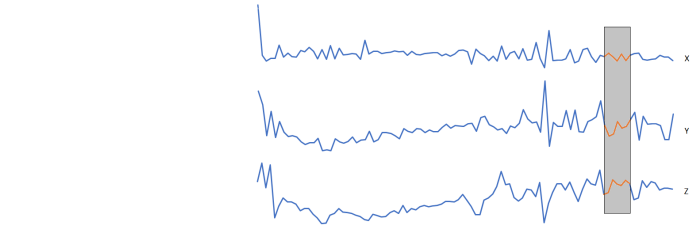
A multivariate subsequence which is the same length as the multivariate shapelet that is passed in is compared using a modified multi-dimensional Euclidean distance. The multivariate shapelet is initially z-normalised along each dimension respectively. Each individual dimension within the subsequence is z-normalised. In 6.4c we demonstrate a shapelet and an extract normalised subsequence. These two sets of series are then paired with there matching dimensions. For each point we calculate the square difference and sum across the whole series. The square root of this summed value is the distance for that particular subsequence compared to the multivariate shapelet. This process continues for the whole series until we find the minimum distance, which is the position of closest match. In this methods because of the nature of the distance calculations we are able to leverage all of the distance early abandoning techniques.

The runtime complexity for the MST_D method is bounded by $O(n^2m^4d)$. This is because we evaluate the same number of shapelets as univariate series of the same size and length. When calculating the distance we perform d operations, in the univariate case of d being equal to 1 this simplifies to the original worst case complexity.

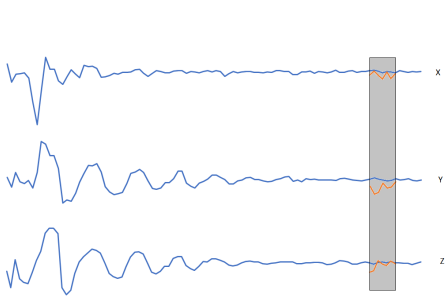
The most important aspect of the MST_D algorithm is that the minimum distance for a multivariate series and a multi-dimensional shapelet is the position of best match is maintained across the channels. The motivation for this method is that for gesture recognition where a particular gesture is performed, all the channels (X, Y and Z) should have information about this event at the same point, but that the phase independence of shapelets means the information can be captured even though it can occur at any time interval.

Algorithm 16 $\text{sDist}_D(\mathbf{MT}, MShapelet, i, m, dimensions, l)$

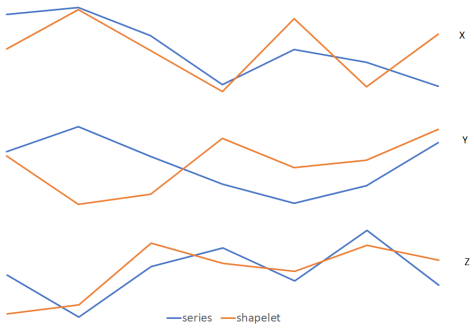
Where MT is a set of time Series.
$min\_dist, dist$
for $p = 1$ **to** $m - l$ **do**
  $sq\_dist\_sum = 0$
  for $d = 1$ **to** $dimensions$ **do**
    $subsequence = \mathbf{MT}^{l}_{i,d,p}$
    $sq\_dist\_sum = sq\_dist\_sum + dist(MShapelet, subsequence)^2$
  $dist = \sqrt{sq\_dist\_sum}$
  if $dist < min\_dist$ **then**
    $min\_dist = dist$
return min_dist



(a) Shapelet extraction



(b) Shapelet matching



(c) Normalised distance

Figure 6.4: An example of extracting a Shapelet_D from a many dimensional series, and comparing it to a different series. Orange is the extracted shapelet, and blue is either the time series the shapelet is extracted from, or being compared too.

6.5.2 Multidimensional Independent Shapelets

The second multivariate shapelet method is called Multidimensional Independent Shapelets (MST_I). This method is similar to MST_D which extracts multi-dimensional shapelets. Each subsequence within the multi-dimensional shapelet finds the minimum distance to its respective dimension independent of the other calculations. MST_D could be considered a special case of the MST_I algorithm where the best independent phase matches coincide.

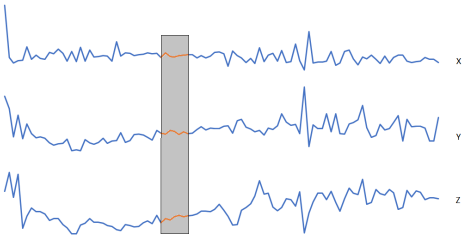
In algorithm 17 we describe the algorithm for calculating the distance between a multivariate shapelet and the the multivariate series passed into the function. The shapelet band that is extracted is shown in 6.5a.

Initially the algorithm loops through each dimension of both the multivariate shapelet and the multivariate time series, each shapelet subsequence from each dimension is slide along its matching dimensional series. This is illustrated in 6.5b where the individual subsequences are disconnected. For each of the dimensions, the best matching location is found. This is where the minimum distance between the shapelet and the subsequence is calculated. The distance is calculated by normalising the subsequence and then calculating the Euclidean distance, we illustrate this in 6.5c. The runtime complexity of this algorithm also scales linearly with the number of dimensions as we essentially perform d number of distance checks for a shapelet, the algorithm is bounded by $O(n^2m^4d)$.

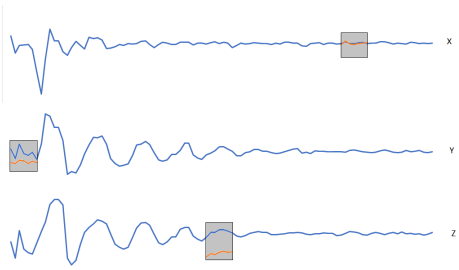
The motivation for this method is that we believe whilst the shapelet extracted is dependent on the features being in phase, the places where they occur in other series could be independent of one another. The ideal case is if there is a small amount of lag in either of the other dimensions. This type of feature is most likely to occur in human activity recognition where a particular set of movements may happen in the same phase and timing, but between many samples there can be slight timing variations across the dimensions. This problem is also in part due to motor redundancy and how a similar movement or gesture can be presented in a infinite number of slight muscular variations within the body.

Algorithm 17 $\text{sDist}_I(\mathbf{MT}, MShapelet, i, m, dimensions, l)$

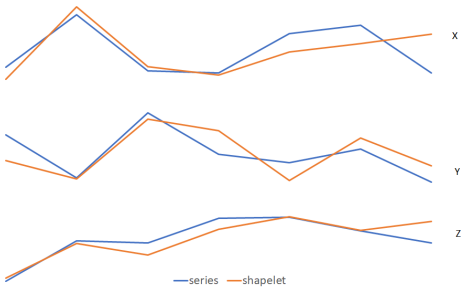
Where MT_i is a time Series.
$min\_dist, dist$
$dist\_sum = 0$
for $d = 1$ **to** $dimensions$ **do**
  $min\_dist = 0$
  for $p = 1$ **to** $m - l$ **do**
    $subsequence = \mathbf{MT}^l_{i,d,p}$
    $dist = distance(MShapelet_d, subsequence)$
    if $dist < min\_dist$ **then**
      $min\_dist = dist$
  $dist\_sum = dist\_sum + min\_dist$
return $dist\_sum$



(a) Shapelet extraction



(b) Shapelet matching



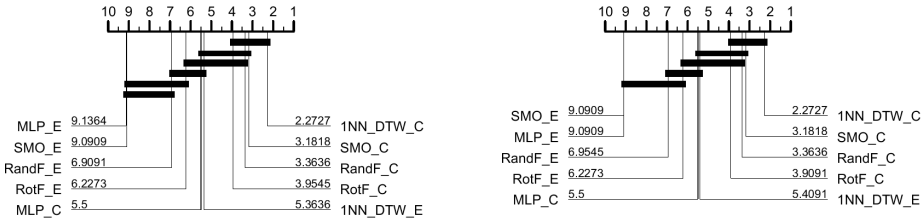
(c) Normalised distance

Figure 6.5: We present an illustrative example of extracting a $\text{Shapelet}_{\text{I}}$ from a many dimensional series, and comparing it to a different series. Orange is the extracted shapelet, and blue is either the time series the shapelet is extracted from, or being compared too.

6.6 Evaluation

The experimental setup follows the same approach outlined in [8]. We perform 100 fold resampling on the data. For each algorithm presented we have performed 2,400 experiments. The data presented in the tables are the mean average accuracy across the 100 folds. The critical difference diagrams are calculated from these mean averages.

We initially present the results for the univariate methods for both, concatenation and dimensional ensembling. These results show how concatenating dimensions into a single univariate series is superior to dimensional ensembling. However there are a few cases where the ensemble approach for



(a) Accuracy

(b) Balanced Accuracy

Figure 6.6: Accuracy and balanced accuracy of 10 algorithms using five simple classifiers. These algorithms are RotationForest(RotF), RandomForest (RandF), Support Vector Machine using a quadratic kernel (SMO), Multi-Layer Perceptron (MLP) and 1 nearest neighbour with dynamic time warping (1NN_DTW). We use the notation _C to denote concatenation, and _E to denote ensembled across dimensions.

DTW out performs concatenation. We believe given a cross-validated training approach to weight the respective ensemble dimensions, the ensembled method would be more robust. However the scale, and time requirements to cross-validate on these datasets is out of the scope of this work. In Figure 6.6 we present two critical difference diagrams, one comparing the average accuracy of the simple classifiers, and one comparing the balanced accuracy.

6.6.1 Shapelets

We present the full enumeration results for our three multivariate shapelet methods on 16 datasets in Table 6.1. These experiments were performed on 100 resamples of the train/test splits to create 100 unique datasets with which we average our summary statistics on.

In Table 6.2 we have performed a fixed one hour run time evaluation of the shapelet space for five additional datasets. The simple worst-case complexity

analysis we conducted earlier, combined with our work in chapter 5 we are able to calculate ahead of time how many shapelets we can evaluate for all three multivariate methods.

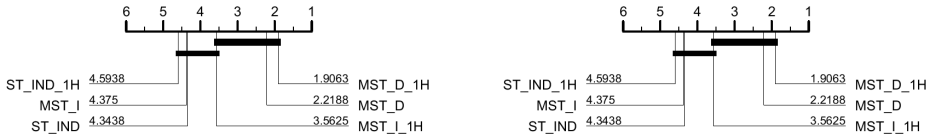
The main aims of this work was to create a comparable Shapelet Transform for multivariate time series classification, that scales better than concatenation, and alleviates most of the complexity and parameter tuning of ensembling. This work also needs to be able to leverage the heuristic techniques previously developed, but also scale well with them too. For contract classification we want to ensure that we can maintain accuracy as the datasets increase in size.

We constrained the number of shapelets that are considered, and fixed the run time of the algorithm. All the datasets were limited to a one hour run time. Alongside this we performed full enumerations for all the datasets where possible. This meant we were able to evaluate an additional five datasets. In Figure 6.7 we present the critical difference diagrams comparing the full enumeration of the multivariate datasets vs. the one hour run time. This is a comparison of only the 16 that the full enumeration could complete. We demonstrate that for both accuracy and balanced accuracy, and with pairwise Wilcoxon and the student t-test there is no significant difference between the 1hour contract approach and the full enumeration. This shows that our contract classifier is currently able to scale on these datasets without issue.

In Table 6.2 we have the results for three multivariate DTW approaches. We compare these three approaches with that of the three shapelets using critical difference diagrams for both accuracy and balanced accuracy, presented in Figure 6.8.

From these results we find that MST_D is not significantly worse than any of the multivariate DTW approaches on 21 datasets all with 100 fold resampling. In addition to the critical difference tests, we performed multiple pairwise tests and show that MST_D with a constrained runtime of one hour is not significantly worse than any of the three DTW multivariate methods.

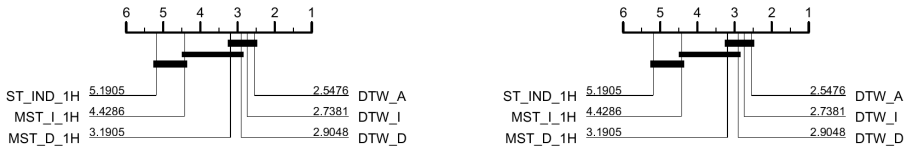
The MST_I method was significantly worse than the DTW_A and DTW_I but was not significantly worse than DTW_D on a Wilcoxon sign ranked test.



(a) Accuracy (b) Balanced Accuracy

Figure 6.7: Accuracy and Balanced Accuracy

Finally we found that the independent shapelet method was significantly worse than all multivariate DTW methods as well as MST_D and was not significantly worse than MST_I.



(a) Accuracy

(b) Balanced Accuracy

Figure 6.8: Two critical difference diagrams comparing the three shapelet algorithms with the three multivariate dynamic time warping algorithms.

datasets	MST_D	ST_IND	MST_I	DTW_A	DTW_D	DTW_I
AALTD_0	0.646 (0.04)	0.583 (0.06)	0.569 (0.05)	0.664 (0.03)	**0.681** (0.03)	0.649 (0.03)
AALTD_1	0.792 (0.03)	0.725 (0.04)	0.744 (0.04)	0.805 (0.03)	0.804 (0.03)	**0.809** (0.03)
AALTD_2	0.608 (0.04)	0.529 (0.05)	0.557 (0.04)	0.667 (0.04)	**0.675 (0.03)**	0.671 (0.04)
AALTD_3	0.661 (0.05)	0.61 (0.05)	0.656 (0.05)	**0.684** (0.04)	0.68 (0.04)	0.683 (0.04)
AALTD_4	0.624 (0.04)	0.576 (0.04)	0.619 (0.04)	0.657 (0.04)	**0.667** (0.04)	**0.667** (0.04)
AALTD_5	0.767 (0.04)	0.735 (0.04)	0.735 (0.04)	0.789 (0.04)	**0.797** (0.04)	0.777 (0.04)
AALTD_6	0.617 (0.04)	0.542 (0.05)	0.461 (0.06)	0.654 (0.04)	**0.671** (0.03)	0.639 (0.03)
AALTD_7	**0.796** (0.04)	0.739 (0.04)	0.746 (0.04)	0.791 (0.03)	0.784 (0.03)	0.791 (0.03)
ArticularyWordLL	0.856 (0.02)	0.828 (0.02)	**0.865** (0.02)	0.84 (0.02)	0.843 (0.02)	0.83 (0.02)
ArticularyWordT1	0.923 (0.02)	0.901 (0.02)	0.894 (0.02)	0.921 (0.01)	**0.924** (0.01)	0.908 (0.01)
ArticularyWordUL	**0.811** (0.03)	0.718 (0.03)	0.829 (0.03)	0.741 (0.02)	0.719 (0.02)	0.749 (0.02)
HandwritingA	0.481 (0.03)	0.442 (0.03)	0.426 (0.03)	0.601 (0.03)	**0.609** (0.02)	0.48 (0.02)
JapaneseVowels	0.887 (0.02)	0.808 (0.03)	0.88 (0.02)	0.957 (0.01)	0.955 (0.01)	**0.959** (0.01)
MVMotionA	**0.979** (0.02)	0.956 (0.02)	0.963 (0.03)	0.912 (0.05)	0.77 (0.04)	0.912 (0.05)
MVMotionAG	0.984 (0.02)	0.953 (0.03)	0.961 (0.03)	**0.999** (0)	0.951 (0.04)	**0.999** (0)
MVMotionG	0.936 (0.04)	0.939 (0.03)	0.933 (0.04)	**0.996** (0.01)	0.917 (0.04)	**0.996** (0.01)
Wins	3	0	1	3	7	5

Table 6.1: A table of results for the Full searches for the three shapelet algorithms, and the three dynamic time warping algorithms

.

datasets	MST_D_1H	ST_IND_1H	MST_I_1H	DTW_A	DTW_D	DTW_I
AALTD_0	0.646 (0.04)	0.583 (0.06)	0.569 (0.05)	0.664 (0.03)	**0.681 (0.03)**	0.649 (0.03)
AALTD_1	0.792 (0.03)	0.725 (0.04)	0.744 (0.04)	0.805 (0.03)	0.804 (0.03)	**0.809 (0.03)**
AALTD_2	0.608 (0.04)	0.529 (0.05)	0.557 (0.04)	0.667 (0.04)	**0.675 (0.03)**	0.671 (0.04)
AALTD_3	0.661 (0.05)	0.61 (0.05)	0.656 (0.05)	**0.684 (0.04)**	0.68 (0.04)	0.683 (0.04)
AALTD_4	0.624 (0.04)	0.576 (0.04)	0.619 (0.04)	0.657 (0.04)	**0.667 (0.04)**	**0.667 (0.04)**
AALTD_5	0.767 (0.04)	0.735 (0.04)	0.735 (0.04)	0.789 (0.04)	**0.797 (0.04)**	0.777 (0.04)
AALTD_6	0.617 (0.04)	0.542 (0.05)	0.461 (0.06)	0.654 (0.04)	**0.671 (0.03)**	0.639 (0.03)
AALTD_7	**0.796 (0.04)**	0.739 (0.04)	0.746 (0.04)	0.791 (0.03)	0.784 (0.03)	0.791 (0.03)
ArticularyWordLL	0.856 (0.02)	0.828 (0.02)	**0.865 (0.02)**	0.84 (0.02)	0.843 (0.02)	0.83 (0.02)
ArticularyWordT1	0.923 (0.02)	0.901 (0.02)	0.894 (0.02)	0.921 (0.01)	**0.924 (0.01)**	0.908 (0.01)
ArticularyWordUL	**0.811 (0.03)**	0.718 (0.03)	0.829 (0.03)	0.741 (0.02)	0.719 (0.02)	0.749 (0.02)
CricketLeft	0.92 (0.03)	0.819 (0.04)	0.869 (0.03)	0.927 (0.02)	**0.933 (0.02)**	0.887 (0.02)
CricketRight	0.935 (0.02)	0.93 (0.03)	0.93 (0.03)	0.939 (0.03)	0.924 (0.03)	**0.945 (0.03)**
Epilepsy	0.969 (0.01)	0.978 (0.02)	**0.981 (0.01)**	0.965 (0.01)	0.957 (0.02)	0.969 (0.01)
HandwritingAccelerometer	0.481 (0.03)	0.442 (0.03)	0.426 (0.03)	0.601 (0.03)	**0.609 (0.02)**	0.48 (0.02)
HandwritingGyroscope	0.84 (0.01)	0.711 (0.1)	0.769 (0.11)	0.861 (0.05)	**0.863 (0.05)**	0.785 (0.04)
JapaneseVowels	0.887 (0.02)	0.808 (0.03)	0.88 (0.02)	0.957 (0.01)	0.955 (0.01)	**0.959 (0.01)**
MVMotionA	**0.979 (0.02)**	0.956 (0.02)	0.963 (0.03)	0.912 (0.05)	0.77 (0.04)	0.912 (0.05)
MVMotionAG	0.984 (0.02)	0.953 (0.03)	0.961 (0.03)	**0.999 (0)**	0.951 (0.04)	**0.999 (0)**
MVMotionG	0.936 (0.04)	0.939 (0.03)	0.933 (0.04)	**0.996 (0.01)**	0.917 (0.04)	**0.996 (0.01)**
UWaveGesture	0.898 (0.02)	0.868 (0.02)	0.862 (0.02)	0.919 (0.01)	**0.925 (0.01)**	0.909 (0.01)
Wins	3	0	2	3	10	6

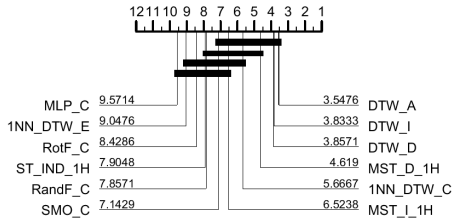
Table 6.2: A table of results showing the results for the one hour runtimes of the three shapelet algorithms using random shapelet selection and the three dynamic time warping algorithms. The standard deviation across the 30 folds is in brackets.

.

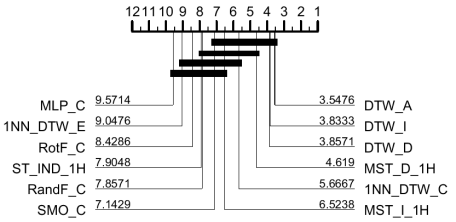
6.6.2 Comparing multivariate approaches with simple classifiers

We compare the three shapelet methods and the three multivariate DTW methods to the initial benchmarks we conducted earlier. In the initial comparison of the simple classifiers we noted that ensembles methods overall performed worse than concatenation approaches, to ensure readability on the critical difference diagrams in Figure 6.9 we have opted only to compare all the concatenation approaches, 1NN_DTW ensemble and the multivariate methods. We present these results on all 22 datasets where we have calculated the accuracy, balanced accuracy, log likelihood and AUROC. On the final analysis of these approaches comparing via log likelihood can give us insight into how the different approaches generate probability distributions for our predictions, and the confidence of true positives and true negatives. We compare using AUROC as a means of understanding the True positive rate and the False positive rate.

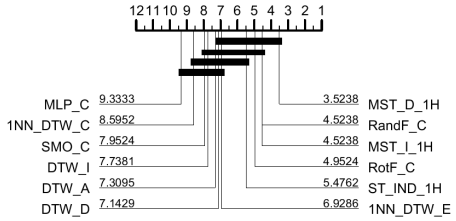
For accuracy and balanced accuracy there is no significant difference between the top 7 approaches on the multivariate datasets, these algorithms are DTW_A, DTW_D, DTW_I, MST_D_1H, 1NN_DTW_C, MST_I_1H and SMO_C. These final results are interesting for a number of reasons. They show that the current state-of-the-art, which is DTW_A, is not significantly better than simpler concatenation approaches and that the perceived wisdom of scaling to multivariate may not be difficult. We constructed an additional critical difference diagram of the top four DTW methods and the three shapelet methods in Figure 6.10 which shows that shapelets are not significantly better than simpler approaches on these problems. However, we stand by the position of concatenated shapelets being an untenable algorithm as series and dimensions increase. Interestingly shapelets has much higher rankings in the AUROC tests we performed and this might indicate that the shapelets we are finding enable good recall and sensitivity in HESCA.



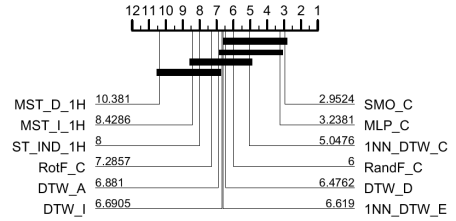
(a) Accuracy



(b) Balanced Accuracy

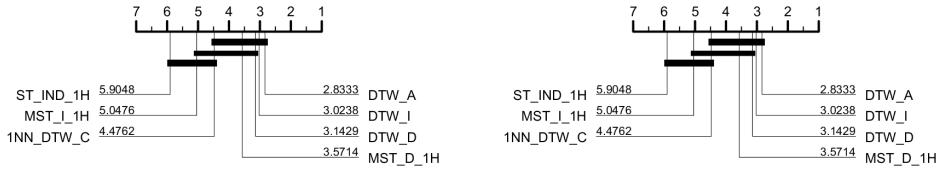


(c) AUROC



(d) Log Likelihood

Figure 6.9: Four critical difference diagrams showing Accuracy, Balanced Accuracy, AUROC and log likelihood of the best 12 algorithms.



(a) Accuracy (b) Balanced Accuracy

Figure 6.10: Two critical difference diagrams showing accuracy and balanced accuracy of the three multivariate DTW algorithms, the three timed shapelet algorithms and 1NN_DTW on concatenated data

6.7 Case Study: MVMotionA

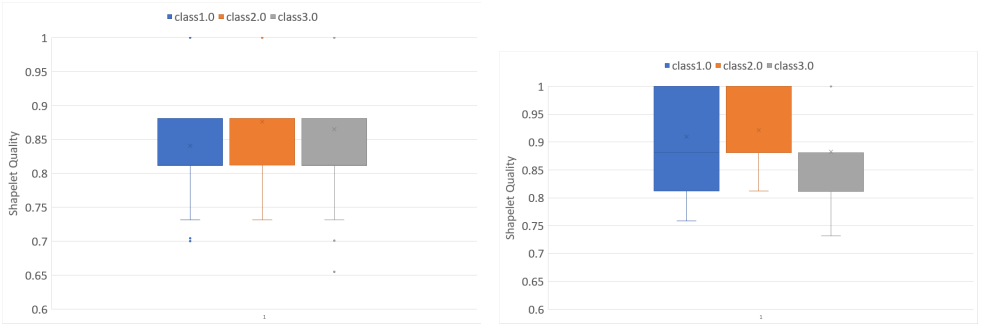
MVMotion is a dataset captured at UEA. It is a human activity recognition problem. The expectation was that the multivariate shapelet based approaches would be well suited in motion based classification. The dataset was introduced in chapter 3 where the method of capturing the accelerometer data was covered in greater detail. The aim of this problem was to detect from a smartphone device whether an individual was sitting, running, walking or playing badminton. In Figure 6.11 the four classes and the X,Y,Z dimensions are shown.

In the experiments performed all three shapelet methods were found to have the highest accuracy on this particular problem. Over the 100 folds all three shapelets methods were at least 5% more accurate compared to the DTW based approaches.

Given the set of shapelets extracted we also wanted to consider the average quality of the shapelets, high quality means that the multivariate shapelet is able to differentiate between the classes well, and so is a good measure of whether it is capturing the underlying structure of the data.

Figure 6.11: Four classes for the MVMotionA dataset

For the MST_D the average shapelet quality of the 300 total shapelets found is 0.86 with a standard deviation of 0.073 and for MST_I is 0.904 with a standard deviation 0.078. In Figure 6.12 the box and whiskers for both MST_D and MST_I are shown. Initially with a high average quality it appears the MST_I should be better, however upon inspection of the individual averages on a class basis, MST_I is able to separate classes 1 and 2 very well, with some shapelets perfectly separating one class from the rest. However, separating class 3 from the rest appears to be more difficult and could explain why the overall average quality does not translate to improved accuracy. Despite the quality overall being better for MST_I, the MST_D has superior accuracy. This may show that despite the shapelet being able to separate the training data well, this property is present in the test set and overall explains why there is some accuracy difference over the 100 folds and more variance with respect to MST_I.



(a) MST_D (b) MST_I

Figure 6.12: Box and Whiskers plots of the quality of shapelets broken down by class

6.8 Conclusion

In conclusion, we have collated a reasonable set of data for multivariate time series classification. Some of which was constructed at UEA, but most of which we collated from the literature. We aimed to convert this into a

common framework of Weka.

We have evaluated a set of algorithms from within the literature that we implemented in Weka and Java. Some of the multivariate algorithms from the literature we were not able to get or sufficiently recreate due to time constraints and scope. We implemented three multivariate dynamic time warping methods, that we verified, alongside a MATLAB implementation.

We then proposed three shapelet methods and modified the existing Shapelet Transform framework to accommodate multivariate time series data. We created a series of simple ensembles, and filtered the multivariate data into a univariate space by concatenating the dimensions. This created 19 different algorithms to evaluate on the 22 datasets. In our previous experiments we established that performing resampling of the datasets can create more robust estimates of the problem's accuracy. We chose to perform 100 fold resamples on the 22 datasets, essentially creating 2200 problems performing 41,800 experiments.

From these experiments we presented initial benchmarking results of the simple classifiers, demonstrating that without any tuning concatenation is an effective method of filtering multivariate datasets. We identified issues with trying to ensemble or concatenate data and use with the Shapelet Transform, and the contract classifier version. The three shapelet methods we proposed were able to find multi-dimensional shapelets whilst only requiring an additional linear component to the worst case runtime. This is a significant improvement over the quartic scaling of when the dimensions are concatenated together.

The multivariate Shapelet Transform with the dependent distance measure, was not significantly worse than any of the other approaches. In thoroughly analysing the Shapelet Transform we also determined the multivariate dynamic warping approaches are not significantly worse than each other, but are also not significantly worse than concatenated dynamic time warping. This result can suggest two possibilities, the datasets we have presented are biased towards dynamic time warping approaches, especially as the Shapelet Transform has been shown to be significantly better than DTW on many problems. The alternative is that scaling to multivariate

problems is not difficult, and univariate classifiers can achieve relatively good accuracy without specialised multivariate models.

Finally, we conducted a small case study on the MVMotionA dataset which we introduced in [17]. This dataset is accelerometer data captured from a smart phone. The case study looked at the shapelets found from within the series. It also looked at the quality of the shapelets on a class basis. We then discussed how this quality can affect the accuracy of a transform, and whether the multivariate shapelets are able to separate classes correctly from one another. We believe the multivariate shapelets capture different features to multivariate dynamic time warping, are more interpretable than concatenated shapelets and will fill a niche area of multivariate time series classification.

Chapter 7

Conclusions and Future Work

The work in this thesis was initially aimed at reducing the runtime of the brute-force search algorithm of the shapelet transform (ST). In time series classification (TSC) shapelet based methods have been shown to be a very effective model for classification [8, 70, 83, 40].

There have been a number of different improvements and heuristic speed ups proposed since the algorithm's inception. We initially wanted to answer the question: can we make the shapelet transform faster without reducing accuracy? The majority of research into shapelet speed ups has consisted of improvements to reducing the number of operations performed when evaluating a single shapelet [114, 84, 83, 79].

However, the main issue with some of these methods is they either used large amounts of additional memory (making it unscalable), or they were only possible on two class problems. The secondary aims of this thesis became apparent once we performed a large scale evaluation of the shapelet methods and had to drastically sample some of the largest datasets to achieve results. The secondary aim of this thesis was, can we give a shapelet transform a time limit to find shapelets, and still achieve comparable accuracy to a full enumeration? We aimed to explore heuristic searches and how they affected

accuracy, with the final aim being to create a multivariate representation of shapelets. Evaluating large multivariate datasets would be all but impossible without the heuristic searches and distance measures created in the earlier work. The findings of this thesis are as follows: the Shapelet Transform when paired with the heterogeneous ensemble of simple classifiers (ST HESCA) is the second best time classification algorithm for univariate time series classification. The state-of-the-art classifier is the collective of transformation ensembles (COTE) which the shapelet transform forms a core part of [8, 6].

7.1 Discussion of Contributions

The first contribution of this thesis was improving the shapelet transform on multi class problems. These improvements were designed to reduce the runtime and improve the accuracy on multiclass problems. We made changes to allow entropy pruning on multi class datasets and we created a new sDist function for evaluating shapelet order which on average increased early abandons. In addition to these speed improvements we developed a balancing mechanism to ensure shapelets from all classes were represented in the transform. In chapter 4 we presented the results from comparing the operation count improvements and the accuracy improvements. These results showed that with more classes the balanced shapelet transform had improved accuracy over the original algorithm. In the worst cases the algorithm presented did not significantly improve accuracy, but was not worse than the original. In the best and averages cases there were improvements and this prompted the inception of the ST HESCA classifier which selects the relevant shapelet transform depending on the dataset.

Performing these initial experiments helped to create a better experimental methodology, which we designed so that we could perform more robust evaluations of time series classifiers. With this new methodology, the aim was to evaluate the state of the art classifiers within time series classification, the focus of the work in second portion of chapter 4 being on shapelet based classifiers and the wider work from the UEA time series classification group published in [8]. The contributions towards this thesis was that we could

fully demonstrate the significance of the improvements created earlier in chapter 4 where the Shapelet Transform now out performs the original on a number of problems, and is significantly more accurate than most time series classification algorithms in the literature. The aim of the work was to also establish the state of other shapelet algorithms, as the number of overlapping datasets did not allow robust comparisons. We found that even with the help of the authors on recreating there algorithms, the results were less favourable for Learn Shapelets and Fast Shapelets. A minor objective of this work was to open source all of the available results and source code, for replication by the time series community (available [5]).

One of the major issues noticed during the large scale experimental work described in chapter 4 was that the shapelet transform was intractable on the largest problems even with the use of specialist HPC equipment and tailored HPC ST algorithms. With the final goal of this thesis to design multivariate shapelet algorithms it became apparent that more drastic time and space reductions would be required as we were still struggling on univariate problems. The aim was to develop heuristic searches that drastically reduce the runtime of the shapelet search without compromising on classification accuracy. Contract classification was proposed, whereby time requirements were imposed on the shapelet search and a number of shapelets were found until the time limit expired. We find that some of these searches required large parametrisation, where searching for the best parameter set would defeat the object of contract classification. The contributions in chapter 5 were designing and evaluating four shapelet search algorithms considering both one hour and one day runtimes and comparing these results with the current ST. We also added an additional dataset to the UCR-UEA archive from the literature [94] which was the largest problem found in the literature for time series classification (HeartbeatBIDMC).

The final contribution in this thesis is one of the first large scale experiments on multivariate TSC datasets. We gathered datasets from the literature and processed them into the ARFF format for use with Weka [43]. Selecting a number of simple classifiers, most of which are constituents of HESCA, we performed concatenation and ensembling over dimensions

to create ten classifiers which we could benchmark these initial datasets on. In addition to these simple classifiers we implemented three types of multivariate dynamic time warping, and we created three new multivariate shapelet transforms. We then benchmarked these algorithms in a similar style to [8] aiming to establish a common set of multivariate problems and to provide a framework within Weka for other algorithms to use. It was found that the shapelet algorithms were not significantly worse than other multivariate time series classification approaches, and on certain datasets outperformed dynamic time warping.

Through this thesis we have significantly improved the classification accuracy of the Shapelet Transform for univariate time series classification. We have demonstrated new heuristic speed up techniques that enable faster searching and evaluation of shapelets. We presented the concept of a contract classifier whereby the runtime of an algorithm can be artificially shortened, and demonstrated that no significant accuracy loss was found despite vast improvements in speed. Finally we introduced a new time representation of shapelets in the multivariate domain, that are significantly faster than naively concatenating or ensembling.

7.2 Future Work and Extensions

The results for the searching and runtime reductions on univariate problems were very promising and demonstrated the effectiveness of heuristic searches. We believe that better heuristic searches could be created that consider less data. One of the major problems is that the search for shapelets is only half of the runtime complexity of the algorithm. The other component is the quality and distance calculations, which requires similar amounts of work. Significant amounts of research has been performed in reducing the number of operations when evaluating a single shapelet, however, we believe that even more drastic speed up algorithms could be proposed. As opposed to the full sliding window function being used, a sufficient number of random evaluations could provide a cheap alternative to establishing tentative distance values. In this way the search space could be quickly pruned using minimal calculations

and then a set of smaller shapelets could be fully evaluated. This idea is similar to fast shapelets in some respects, and we did present the results of a fast shapelet transform in [18]. However, we believe PAA and SAX destroy the subtle shape information that is an important part of what makes shapelets a good representation. One potential future area of research is in creating contract classifiers of the other leading time series classification algorithms, a contract COTE being the final aim.

The novel contribution of this work is in the multivariate domain. The work presented in chapter 6 was one of the first large scale analyses of multivariate datasets, where most studies had focused on a handful of datasets. These 23 datasets should provide the foundation for additional research and more data will hopefully be contributed by the community. We had hoped to provide a decisive multivariate shapelet transform that outperformed all other approaches, however, this was not the case. Further work would ideally focus on looking at the problems with the current multivariate shapelet representations and consider how to avoid some of them. One of these problems is that currently the multivariate shapelets are attempting to find bands of shapelets in the same phase across dimensions. If there is any lag in the activity the length of the whole shapelet band needs to increase to accommodate this, and results in the individual dimension sequences containing noise. As this signal to noise ratio increases the likelihood of a good shapelet match will decrease and so the true shapelet may be missed. One possibility is that the multivariate shapelets phase could be independent of each other, combining the shapelets from the same series and across dimensions. This is a much more difficult problem than just finding bands of multivariate shapelets, especially as the interpretability of shapelets may decrease which is one of their beneficial properties.

Finally having created a multivariate shapelet transform we expect other major TSC algorithms to consider multivariate representations with changes to the BOSS algorithm and potentially a new multivariate elastic ensemble, or time series forest [94, 69, 30]. The culmination of this would be a multivariate COTE, converting the state-of-the-art univariate time series classifier into the state-of-the-art multivariate classifier.

Appendices

Table 1: The average accuracies for the Shapelet Transform, Learn Shapelets and Fast Shapelets averaged over a 100 resamples for the 85 UCR datasets

Datasets	ST_HESCA	LS	FS
Adiac	**0.768**	0.527	0.555
ArrowHead	**0.851**	0.841	0.675
Beef	**0.736**	0.698	0.502
BeetleFly	**0.874**	0.861	0.795
BirdChicken	**0.927**	0.863	0.862
Car	**0.902**	0.856	0.736
CBF	**0.986**	0.977	0.924
ChlorineConcentration	**0.682**	0.586	0.566
CinCECGtorso	**0.918**	0.855	0.741
Coffee	**0.995**	0.995	0.917
Computers	**0.785**	0.654	0.5
CricketX	**0.777**	0.744	0.479
CricketY	**0.762**	0.726	0.509
CricketZ	**0.798**	0.754	0.466
DiatomSizeReduction	0.911	**0.927**	0.873
DistalPhalanxOutlineCorrect	**0.829**	0.822	0.78
DistalPhalanxOutlineAgeGroup	**0.819**	0.81	0.745
DistalPhalanxTW	**0.69**	0.659	0.623
Earthquakes	0.737	0.742	**0.747**
ECG200	0.84	**0.871**	0.806
ECG5000	**0.943**	0.94	0.922
ECGFiveDays	0.955	0.985	**0.986**
ElectricDevices	**0.895**	0.709	0.262
FaceAll	**0.968**	0.926	0.772
FaceFour	0.794	**0.957**	0.869
FacesUCR	0.909	**0.939**	0.701
FiftyWords	**0.713**	0.694	0.512
Fish	**0.974**	0.94	0.742
FordA	**0.965**	0.895	0.785
FordB	**0.915**	0.89	0.783
GunPoint	**0.999**	0.983	0.93
Ham	0.808	**0.832**	0.677

Continued on next page

Table 1 – continued from previous page

Datasets	ST	LS	FS
HandOutlines	**0.924**	0.837	0.841
Haptics	**0.512**	0.478	0.356
Herring	**0.653**	0.628	0.558
InlineSkate	**0.393**	0.299	0.257
InsectWingbeatSound	**0.617**	0.55	0.488
ItalyPowerDemand	**0.953**	0.952	0.909
LargeKitchenAppliances	**0.933**	0.765	0.419
Lightning2	0.659	**0.759**	0.48
Lightning7	0.724	**0.765**	0.101
Mallat	**0.972**	0.951	0.893
Meat	**0.966**	0.814	0.924
MedicalImages	0.691	**0.704**	0.609
MiddlePhalanxOutlineCorrect	0.815	**0.822**	0.716
MiddlePhalanxOutlineAgeGroup	**0.694**	0.679	0.613
MiddlePhalanxTW	**0.579**	0.54	0.519
MoteStrain	**0.882**	0.876	0.793
NonInvasiveFatalECGThorax1	**0.947**	0.6	0.71
NonInvasiveFatalECGThorax2	**0.954**	0.739	0.758
OliveOil	**0.881**	0.172	0.765
OSULeaf	**0.934**	0.771	0.679
PhalangesOutlinesCorrect	**0.794**	0.783	0.73
Phoneme	**0.329**	0.152	0.173
Plane	**1**	0.995	0.97
ProximalPhalanxOutlineCorrect	**0.881**	0.793	0.797
ProximalPhalanxOutlineAgeGroup	**0.841**	0.832	0.797
ProximalPhalanxTW	**0.803**	0.794	0.716
RefrigerationDevices	**0.761**	0.642	0.574
ScreenType	**0.676**	0.445	0.365
ShapeletSim	0.934	0.933	**1**
ShapesAll	**0.854**	0.76	0.598
SmallKitchenAppliances	**0.802**	0.663	0.333
SonyAIBORobotSurface1	0.888	0.906	**0.918**
SonyAIBORobotSurface2	**0.924**	0.9	0.849
StarlightCurves	**0.977**	0.888	0.908

Continued on next page

Table 1 – continued from previous page

Datasets	ST	LS	FS
Strawberry	**0.968**	0.925	0.917
SwedishLeaf	**0.939**	0.899	0.758
Symbols	0.862	**0.919**	0.908
SyntheticControl	0.987	**0.995**	0.92
ToeSegmentation1	**0.954**	0.934	0.904
ToeSegmentation2	**0.947**	0.943	0.873
Trace	**1**	0.996	0.998
TwoLeadECG	0.984	**0.994**	0.92
TwoPatterns	0.952	**0.994**	0.696
UWaveGestureLibraryX	**0.806**	0.804	0.694
UWaveGestureLibraryY	**0.737**	0.718	0.591
UWaveGestureLibraryZ	**0.747**	0.737	0.638
UWaveGestureLibraryAll	**0.942**	0.68	0.766
Wafer	**1**	0.996	0.981
Wine	**0.926**	0.524	0.794
WordSynonyms	**0.582**	0.581	0.461
Worms	**0.719**	0.642	0.622
WormsTwoClass	**0.779**	0.736	0.706
Yoga	0.823	**0.833**	0.721
Wins	71	10	4

Table 2: Two tables presenting a comparison of the overlapping fold 0 datasets and the old ST results presented in [70].

Datasets	old_ST	ST_HESCA
datasets	Old	New
Adiac	0.57	**0.78**
ArrowHead	**0.77**	0.74
Beef	0.83	**0.90**
BeetleFly	**0.75**	0.60
BirdChicken	0.75	**0.80**
CBF	**1.00**	0.97
Car	0.73	**0.92**
ChlorineConcentration	0.70	0.70

Continued on next page

Table 2 – continued from previous page

Datasets	old_ST	ST_HESCA
CinCECGtorso	0.85	**0.95**
Coffee	**1.00**	0.96
Computers	0.70	**0.74**
CricketX	**0.78**	0.77
CricketY	0.76	**0.78**
CricketZ	0.77	**0.79**
DiatomSizeReduction	0.88	**0.92**
DistalPhalanxOutlineAgeGroup	0.74	**0.77**
DistalPhalanxOutlineCorrect	0.74	**0.78**
DistalPhalanxTW	0.63	**0.66**
ECGFiveDays	**1.00**	0.98
Earthquakes	0.73	**0.74**
FaceAll	0.74	**0.78**
FaceFour	**0.94**	0.85
FacesUCR	0.91	0.91
FordA	0.93	**0.97**
FordB	0.79	**0.81**
GunPoint	0.98	**1.00**
Haptics	0.48	**0.52**
Herring	0.67	0.67
InlineSkate	**0.39**	0.37
ItalyPowerDemand	0.95	0.95
LargeKitchenAppliances	**0.88**	0.86
Lightning2	0.66	**0.74**
Lightning7	**0.74**	0.73
MALLAT	0.94	**0.96**
MedicalImages	0.60	**0.67**
MiddlePhalanxOutlineAgeGroup	0.63	**0.64**
MiddlePhalanxOutlineCorrect	0.73	**0.79**
MiddlePhalanxTW	**0.54**	0.52
MoteStrain	0.89	**0.90**
NonInvasiveFetalECGThorax1	0.90	**0.95**
NonInvasiveFetalECGThorax2	0.90	**0.95**
OSULeaf	0.71	**0.97**

Continued on next page

Table 2 – continued from previous page

Datasets	old_ST	ST_HESCA
OliveOil	0.90	0.90
PhalangesOutlinesCorrect	0.75	**0.76**
Plane	1.00	1.00
ProximalPhalanxOutlineAgeGroup	**0.85**	0.84
ProximalPhalanxOutlineCorrect	**0.90**	0.88
ProximalPhalanxTW	0.77	**0.80**
RefrigerationDevices	0.56	**0.58**
ScreenType	**0.53**	0.52
ShapeletSim	0.92	**0.96**
SmallKitchenAppliances	0.77	**0.79**
SonyAIBORobotSurface1	**0.93**	0.86
SonyAIBORobotSurface2	0.88	**0.93**
StarLightCurves	0.98	0.98
SwedishLeaf	0.91	**0.93**
Symbols	**0.89**	0.88
SyntheticControl	0.98	0.98
ToeSegmentation1	0.96	**0.96**
ToeSegmentation2	0.85	**0.91**
Trace	0.98	**1.00**
TwoLeadECG	1.00	1.00
TwoPatterns	0.94	**0.95**
UWaveGestureLibraryX	0.78	**0.80**
UWaveGestureLibraryY	0.70	**0.73**
UWaveGestureLibraryZ	0.73	**0.75**
WordSynonyms	**0.60**	0.57
Worms	0.70	**0.74**
WormsTwoClass	0.77	**0.83**
Yoga	0.80	**0.82**
Fiftywords	**0.72**	0.71
Fish	0.98	**0.99**
Wafer	1.00	1.00

Chapter 8

Bibliography

[1] Ahmed Al-Jawad, Miguel Reyes Adame, Michailas Romanovas, Markus Hobert, Walter Maetzler, Martin Traechtler, Knut Moeller, and Yiannos Manoli. Using multi-dimensional dynamic time warping for tug test instrumentation with inertial sensors. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 212–218. IEEE, 2012.

[2] Fevzi Alimoğlu and Ethem Alpaydin. Combining multiple representations for pen-based handwritten digit recognition. *Turkish Journal of Electrical Engineering & Computer Sciences*, 9(1):1–12, 2001.

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

[4] A. Bagnall, A. Bostrom, and J. Lines. The UEA TSC codebase. `https://bitbucket.org/TonyBagnall/time-series-classification`, .

[5] A. Bagnall, J. Lines, A. Bostrom, and E. Keogh. The UCR/UEA TSC archive. `http://timeseriesclassification.com`, .

[6] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification

with COTE: The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27:2522–2535, 2015.

[7] A. Bagnall, A. Bostrom, J. Large, and J.Lines. Simulated data experiments for time series classification part 1: Accuracy comparison with default settings. Technical report, School of Computing Sciences, University of East Anglia, 2016.

[8] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advance. *Data Mining and Knowledge Discovery*, pages 1–55, 2016.

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

[10] Muzaffar Bashir and Jürgen Kempf. Reduced dynamic time warping for handwriting recognition based on multidimensional time series of a novel pen device. *International Journal of Intelligent Systems and Technologies, WASET*, 3(4):194, 2008.

[11] M. Baydogan and G. Runger. Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery*, 30(2):476–509, 2016.

[12] M. Baydogan, G. Runger, and E. Tuv. A bag-of-features framework to classify time series. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):2796–2802, 2013.

[13] Mustafa Gokce Baydogan and George Runger. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, 29(2):400–422, 2015.

[14] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36(2):23–26, 2006.

[15] A. Bostrom and A. Bagnall. Binary shapelet transform for multiclass time series classification. *Proc. 17th International Conference on Big Data Analytics and Knowledge Discovery (DAWAK)*, 2015.

[16] A. Bostrom and A. Bagnall. Binary shapelet transform for multiclass time series classification. *Transactions on Large-Scale Data and Knowledge Centered Systems XXXII: Special Issue on Big Data Analytics and Knowledge Discovery*, pages 24–46, 2017.

[17] A. Bostrom and A. Bagnall. A Shapelet Transform for Multivariate Time Series Classification. *ArXiv e-prints*, 2017.

[18] A. Bostrom, A. Bagnall, and J. Lines. Evaluating improvements to the shapelet transform. *Knowledge Discovery and Data Mining, in Workshop on Mining and Learning from Time Series*, 2016.

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

[20] Jeremy Buhler and Martin Tompa. Finding motifs using random projections. *Journal of computational biology*, 9(2):225–242, 2002.

[21] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems*, 27(2):188–228, 2002.

[22] L. Chen and R. Ng. On the marriage of Lp-norms and edit distance. In *Proc. 30th International Conference on Very Large Databases (VLDB)*, 2004.

[23] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The UEA-UCR time series classification archive. `http://www.cs.ucr.edu/~eamonn/time_series_data/`, 2015.

[24] Yanping Chen, Bing Hu, Eamonn Keogh, and Gustavo EAPA Batista. Dtw-d: time series semi-supervised learning from a single example. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 383–391. ACM, 2013.

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

[26] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[27] Marco Cuturi. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 929–936, 2011.

[28] Rodrigo Fernandes de Mello and Iker Gondra. Multi-dimensional dynamic time warping for image texture similarity. In *Brazilian Symposium on Artificial Intelligence*, pages 23–32. Springer, 2008.

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

[30] H. Deng, G. Runger, E. Tuv, and M. Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239: 142–153, 2013.

[31] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[32] Miikka Ermes, Juha Pärkkä, Jani Mäntyjärvi, and Ilkka Korhonen. Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. *IEEE transactions on information technology in biomedicine*, 12(1):20–26, 2008.

[33] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. International Conference on Machine Learning*, volume 96, pages 148–156, 1996.

[34] Mohamed F Ghalwash and Zoran Obradovic. Early classification of multivariate temporal observations by extraction of interpretable shapelets. *BMC bioinformatics*, 13(1):1, 2012.

[35] Mohamed F Ghalwash, Vladan Radosavljevic, and Zoran Obradovic. Extraction of interpretable multivariate patterns for early diagnostics. In *2013 IEEE 13th International Conference on Data Mining*, pages 201–210. IEEE, 2013.

[36] Nicholas Gillian, Benjamin Knapp, and Sile O'Modhrain. Recognition of multivariate temporal musical gestures using n-dimensional dynamic time warping. In *NIME*, pages 337–342, 2011.

[37] Fred Glover. *Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges*, pages 1–75. Springer US, 1997.

[38] Daniel Gordon, Danny Hendler, and Lior Rokach. Fast randomized model generation for shapelet-based time series classification. *CoRR*, abs/1209.5038, 2012. URL `http://arxiv.org/abs/1209.5038`.

[39] Tomasz Górecki and Maciej Łuczak. Multivariate time series classification with parametric derivative dynamic time warping. *Expert Systems with Applications*, 42(5):2305–2312, 2015.

[40] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Learning time-series shapelets. In *Proc. 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.

[41] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme. Fast classification of univariate and multivariate time series through shapelet discovery. *Fast classification of univariate and multivariate time series through shapelet discovery*, 49:429–454, 2016.

[42] M Pamela Griffin and J Randall Moorman. Toward the early diagnosis of neonatal sepsis and sepsis-like illness using novel heart rate analysis. *Pediatrics*, 107(1):97–104, 2001.

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

[44] Nacereddine Hammami and Mokhtar Sellam. Tree distribution classifier for automatic spoken arabic digit recognition. In *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*, pages 1–4. IEEE, 2009.

[45] Bastian Hartmann and Norbert 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.

[46] J. Hills. *Mining Time-series Data using Discriminative Subsequences*. PhD thesis, School of Computing Sciences, University of East Anglia, 2015.

[47] 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.

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

[49] Charles AR Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.

[50] Lu Hou, James T Kwok, and Jacek M Zurada. Efficient learning of timeseries shapelets. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

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

[52] Nimish Kale, Jaeseong Lee, Reza Lotfian, and Roozbeh Jafari. Impact of sensor misplacement on dynamic time warping based human activity recognition using wearable computers. In *Proceedings of the conference on Wireless Health*, page 7. ACM, 2012.

[53] Isak Karlsson. *Order in the random forest*. PhD thesis, Department of Computer and Systems Sciences, Stockholm University, 2017.

[54] Isak Karlsson, Panagiotis Papapetrou, and Lars Asker. Multi-channel ecg classification using forests of randomized shapelet trees. In *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, page 43. ACM, 2015.

[55] Isak Karlsson, Panagotis Papapetrou, and Henrik Boström. Forests of randomized shapelet trees. In *International Symposium on Statistical Learning and Data Sciences*, pages 126–136. Springer, 2015.

[56] Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. Early random shapelet forest. In *International Conference on Discovery Science*, pages 261–276. Springer, 2016.

[57] Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. Generalized random shapelet forests. *Data Mining and Knowledge Discovery*, 30(5):1053–1085, 2016.

[58] Juha Kela, Panu Korpipää, Jani Mäntyjärvi, Sanna Kallio, Giuseppe Savino, Luca Jozzo, and Sergio Di Marca. Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299, 2006.

[59] 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.

[60] E. Keogh and M. Pazzani. Derivative dynamic time warping. In *Proc. 1st SIAM International Conference on Data Mining (SDM)*, 2001.

[61] Ming Hsiao Ko, Geoff West, Svetha Venkatesh, and Mohan Kumar. Online context recognition in multisensor systems using dynamic time warping. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on*, pages 283–288. IEEE, 2005.

[62] William H Kruskal. A nonparametric test for the several sample problem. *The Annals of Mathematical Statistics*, pages 525–540, 1952.

[63] Mineichi Kudo, Jun Toyama, and Masaru Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11):1103–1111, 1999.

[64] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials*, 15(3):1192–1209, 2013.

[65] J. Large, J. Lines, and A. Bagnall. The Heterogeneous Ensembles of Standard Classification Algorithms (HESCA): the Whole is Greater than the Sum of its Parts. 2017.

[66] M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.

[67] J. Lin, E. Keogh, W. Li, and S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007.

[68] 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.

[69] J. Lines and A. Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29: 565–592, 2015.

[70] J. Lines, L. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. In *Proc. the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

[71] Jason Lines. *Time Series classification through transformation and ensembles*. PhD thesis, University of East Anglia, 2015.

[72] Jason Lines and Anthony Bagnall. Alternative quality measures for time series shapelets. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 475–483. Springer, 2012.

[73] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[74] E. A. Maharaj. Clusters of time series. *Journal of Classification*, 17: 297–314, 2000.

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

[76] Amy McGovern, Derek H Rosendahl, Rodger A Brown, and Kelvin K Droegemeier. Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction. *Data Mining and Knowledge Discovery*, 22(1-2):232–258, 2011.

[77] Karl Øyvind Mikalsen, Filippo Maria Bianchi, Cristina Soguero-Ruiz, Stein Olav Skrøvseth, Rolv-Ole Lindsetmo, Arthur Revhaug, and Robert Jenssen. Learning similarities between irregularly sampled short multivariate time series from ehrs. 2016.

[78] Alexander McFarlane Mood. Introduction to the theory of statistics. 1950.

[79] A. Mueen, E. Keogh, and N. Young. Logical-shapelets: An expressive primitive for time series classification. In *Proc. 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011.

[80] Om Prasad Patri, Rajgopal Kannan, Anand V Panangadan, and Viktor K Prasanna. Multivariate time series classification using interleaved shapelets. In *NIPS 2015 Time Series Workshop*, 2015.

[81] John C Platt. 12 fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, pages 185–208, 1999.

[82] J Ross Quinlan. C4.5: Programming for machine learning. *Morgan Kauffmann*, 38, 1993.

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

[84] T. Rakthanmanon, J. Bilson, L. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Transactions on Knowledge Discovery from Data*, 7(3), 2013.

[85] C. Ratanamahatana and E. Keogh. Three myths about dynamic time warping data mining. In *Proc. 5th SIAM International Conference on Data Mining (SDM)*, 2005.

[86] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 506–510. SIAM, 2005.

[87] Atif Raza and Stefan Kramer. Ensembles of randomized time series shapelets provide improved accuracy while reducing computational costs. *arXiv preprint arXiv:1702.06712*, 2017.

[88] Xavier Renard, Maria Rifqi, Walid Erray, and Marcin Detyniecki. Random-shapelet: an algorithm for fast shapelet discovery. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE, 2015.

[89] Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1619–1630, 2006.

[90] Dennis W Ruck, Steven K Rogers, Matthew Kabrisky, Mark E Oxley, and Bruce W Suter. The multilayer perceptron as an approximation to

a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.

[91] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[92] Yasushi Sakurai, Spiros Papadimitriou, and Christos Faloutsos. Braid: Stream mining through group lag correlations. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 599–610. ACM, 2005.

[93] P. Schäfer. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6): 1505–1530, 2015.

[94] P. Schäfer. Scalable time series classification. *Data Mining and Knowledge Discovery*, 30(5):1273–1298, 2016.

[95] P. Senin and S. Malinchik. SAX-VSM: interpretable time series classification using sax and vector space model. In *Proc. 13th IEEE International Conference on Data Mining (ICDM)*, 2013.

[96] T Shajina and P Bagavathi Sivakumar. Human gait recognition and classification using time series shapelets. In *Advances in Computing and Communications (ICACC), 2012 International Conference on*, pages 31–34. IEEE, 2012.

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

[98] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 289–297. SIAM, 2015.

[99] Mohammad Shokoohi-Yekta, Bing Hu, Hongxia Jin, Jun Wang, and Eamonn Keogh. Generalizing dtw to the multi-dimensional case requires an adaptive approach. *Data Mining and Knowledge Discovery*, 31(1): 1–31, 2017.

[100] P. Smyth. Clustering sequences with hidden markov models. In *Advances in Neural Information Processing Systems*, volume 9, 1997.

[101] A. Stefan, V. Athitsos, and G. Das. The Move-Split-Merge metric for time series. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1425–1438, 2013.

[102] Jiuqiang Tang and Roger B Dannenberg. Extracting commands from gestures: Gesture spotting and recognition for real-time music performance. In *International Symposium on Computer Music Modeling and Retrieval*, pages 72–85. Springer, 2013.

[103] Gineke A ten Holt, Marcel JT Reinders, and EA Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In *Thirteenth annual conference of the Advanced School for Computing and Imaging*, volume 300, 2007.

[104] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1): 91–108, 2005.

[105] Liudmila Ulanova, Nurjahan Begum, and Eamonn Keogh. Scalable clustering of time series with u-shapelets. In *SIAM international conference on data mining (SDM 2015)*. SIAM, 2015.

[106] José Ramón Villar, Silvia González, Javier Sedano, Camelia Chira, and José M Trejo. Human activity recognition and feature selection for stroke early diagnosis. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 659–668. Springer, 2013.

[107] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

[108] Li Wei, Eamonn Keogh, and Xiaopeng Xi. Saxually explicit images: Finding unusual shapes. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 711–720. IEEE, 2006.

[109] Xiaoqing Weng and Junyi Shen. Classification of multivariate time series using locality preserving projections. *Knowledge-Based Systems*, 21(7):581–587, 2008.

[110] Xiaoqing Weng and Junyi Shen. Classification of multivariate time series using two-dimensional singular value decomposition. *Knowledge-Based Systems*, 21(7):535–539, 2008.

[111] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *Trans. Evol. Comp*, 1(1):67–82, 1997.

[112] Zhengzheng Xing, Jian Pei, S Yu Philip, and Ke Wang. Extracting interpretable features for early classification on time series. In *SDM*, volume 11, pages 247–258. SIAM, 2011.

[113] Zhengzheng Xing, Jian Pei, and S Yu Philip. Early classification on time series. *Knowledge and information systems*, 31(1):105–127, 2012.

[114] 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.

[115] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956. ACM, 2009.

[116] Jesin Zakaria, Abdullah Mueen, Eamonn Keogh, and Neal Young. Accelerating the discovery of unsupervised-shapelets. *Data mining and knowledge discovery*, 30(1):243–281, 2016.

[117] Qiang Zhu and Eamonn Keogh. Mother fugger: mining historical manuscripts with local color patches. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 699–708. IEEE, 2010.