A Shapelet Transform for Time Series Classification

Jason Lines
University of East Anglia
Norwich, Norfolk
United Kingdom
j.lines@uea.ac.uk

Luke M. Davis
University of East Anglia
Norwich, Norfolk
United Kingdom
luke.davis@uea.ac.uk

Jon Hills
University of East Anglia
Norwich, Norfolk
United Kingdom
j.hills@uea.ac.uk

Anthony Bagnall
University of East Anglia
Norwich, Norfolk
United Kingdom
anthony.bagnall@uea.ac.uk

ABSTRACT

The problem of time series classification (TSC), where we consider any real-valued ordered data a time series, presents a specific machine learning challenge as the ordering of variables is often crucial in finding the best discriminating features. One of the most promising recent approaches is to find shapelets within a data set. A shapelet is a time series subsequence that is identified as being representative of class membership. The original research in this field embedded the procedure of finding shapelets within a decision tree. We propose disconnecting the process of finding shapelets from the classification algorithm by proposing a shapelet transformation. We describe a means of extracting the k best shapelets from a data set in a single pass, and then use these shapelets to transform data by calculating the distances from a series to each shapelet. We demonstrate that transformation into this new data space can improve classification accuracy, whilst retaining the explanatory power provided by shapelets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

time series, shapelet, filter, transformation

1. INTRODUCTION

The problem of time series classification (TSC), in which we consider any real-valued ordered data a time series, presents a specific machine learning challenge as the ordering of variables is often crucial in finding the best discriminating features. Until recently, the vast majority of data mining TSC research has focused on alternative distance measures for 1-Nearest Neighbour (1-NN) classifiers based on either the raw data, or on compressions or smoothing of the raw data (see [7] for a comprehensive summary). Despite the evidence in favour of 1-NN classifiers with Euclidean or Dynamic Time Warping (DTW) distance, there has been a spate of recent research proposing alternative approaches. These include shapelets [13, 20, 19], weighted dynamic time warping [10], support vector machines built on variable intervals [16], tree based ensembles constructed on summary statistics [6], fusion of alternative distance measures [2], and transform-based ensembles [1]. We consider the shapelet approach one of the most promising of these new methods.

A shapelet is a time series subsequence that is representative of class membership. The initial shapelet-based classification algorithm is proposed in [19]. The algorithm constructs a decision tree classifier by recursively searching for a discriminatory shapelet on data split via an information gain measure, calculated on the distance to the branching shapelet. The distance between a shapelet and a full series is defined as the closest subsequence in the series to the shapelet, as measured by the normalised Euclidean distance. This was extended by the research presented in [13], where they describe a form of oblique/multi-variate decision tree that uses a branching condition based on conjunctions or disjunctions of shapelets. Shapelets have also been used in futher applications, such as early classification [18] and gesture recognition [8].

Classification with shapelets offers several benefits over competing approaches. Firstly, shapelets are directly interpretable and can offer explanatory insights into the problem domain. Secondly, the shapelet classifier is more compact than many of the alternatives, and hence classifying new instances is faster. Thirdly, shapelets allow for the detection of phase-independent shape-based similarity of subsequences. This type of similarity is often hard to detect with algorithms based on whole series.

In both [20] and [13], the shapelet algorithm is embedded within the decision tree classifier. Whilst decision trees are highly interpretable, they are often outperformed by other classifiers and have a tendency to overfit unless post-pruned or used with a conservative stopping condition. Furthermore, the recursive nature of the decision tree algorithm means that the relatively time-consuming shapelet detection method is called repeatedly. Techniques that significantly speed up the

KDD'12, August 12–16, 2012, Beijing, China.
Copyright 2012 ACM 978-1-4503-1462-6 /12/08 ...$15.00.

shapelet search are described in [13], but it is still a costly operation, especially with multi-class problems.

In [1] we show that transforming a TSC problem into an alternative data space prior to classification can provide a greater level of improvement than developing classifier refinements. Hence, we propose a shapelet transform that creates a new classification data set independently of the classifier. By doing so, we wish to increase classification accuracy while reducing training time and maintaining the interpretability of the model. The essence of the technique is to find the best k shapelets and then derive a new attribute for each shapelet, where the attribute value represents the distance from each case to a shapelet. In doing so, we have introduced several alterations to the shapelet algorithm. Our contributions can be summarised as follows:

1. We introduce a caching algorithm to store the k best shapelets from a data set in a single pass.
2. We propose an alternative shapelet evaluation method that is easier to use with multi-class problems.
3. We evaluate our proposed algorithms and compare them with the tree-based shapelet algorithm.

The paper is structured as follows. Section 2 provides background into time series classification and summarises the shapelet algorithms proposed in [19, 20]. In Section 3, we describe the changes we have made to the shapelet algorithm and propose a way of generating a shapelet transform. In Section 4, we describe our experimental design, and in Section 5, we evaluate the proposed algorithms. Finally, in Section 6, we form our conclusions and discuss future work.

2. BACKGROUND

2.1 Time Series Classification

A time series is a sequence of data that is typically recorded in temporal order at fixed intervals of time. For the problem of time series classification, suppose we have a set of n time series, $\mathbf{T} = \{T_1, T_2, ..., T_n\}$, where each time series has m ordered real-valued observations $T_i =< t_{i1}, t_{i2}, ..., t_{im} >$ and a class value c_i. The objective is to find a function that maps from the space of possible time series to the space of possible class values. Note, we assume for simplicity that all series are the same length.

Classification is a widely explored problem in machine learning and, to some extent, all classification problems rely on a measure of similarity between data. What makes time series classification a distinct and interesting area of investigation is that similarity between series is often embedded within the autocorrelation structure of the data. There are three general approaches to measuring similarity between time series: similarity in time (i.e. correlation-based), similarity in shape (shape-based) and similarity in change (autocorrelation-based). A fuller discussion can be found in the literature [11, 12]. The majority of research has focused on similarity in time or shape. For shape-based similarity, it is common to use elastic measures such as DTW in conjunction with an instance based classifier. Whilst this approach has been shown to work well in a wide variety of domains, using an elastic measure on the whole series means the discriminatory shape can be masked by noise. Shapelets offer a mechanism for detecting phase-independent shape-based similarity of subsequences, and hence represent a better solution to a class of shape-based similarity problems than global elastic measures.

2.2 Shapelets

Shapelets are a time series data mining primitive that can be used to determine similarity based on small common shapes occurring at any point in a series. Finding a shapelet requires generating a set of candidates, defining a distance measure between a shapelet and each time series, and defining a measure of the discriminatory power of a shapelet.

2.2.1 Generating Candidates

A subsequence S of length l of a time series T of length m is a contiguous sequence of l points in T, where $l \leq m$. Any time series of length m contains $(m - l) + 1$ distinct subsequences of length l. We denote the set of all subsequences of length l for series T_i to be $W_{i,l}$ and the set of all subsequences of length l for data set T to be

$$W_l = \{W_{1,l}, \dots, W_{n,l}\}.$$

The set of all candidate shapelets for data set T is

$$W = \{W_{min}, W_{min+1}, \dots, W_{max}\},$$

where $min \geq 3$ and $max \leq m$. Note that W is very large, with $O(m^3)$ candidate shapelets. The majority of the research in [19, 20, 13] relates to the efficient pruning of W to improve the time complexity of the exhaustive search. Our main focus, as described in Section 3, is using shapelets more effectively for classification. The generic shapelet finding algorithm is defined in Algorithm 1.

Algorithm 1 ShapeletSelection (T, min,max)

```
best = 0;
bestShapelet = ∅;
C = classLabels(T);
W = generateCandidates(T, min, max);
for l = min to max do
    for all subsequence S in W_l do
        D_S = findDistances(S, W_l);
        quality = assessCandidate(S, D_S);
        if quality > best then
            best = quality;
            bestShapelet = S;
        end if
    end for
end for
return bestShapelet;
```

Note that we independently normalise all elements of W. This is justified because we are interested in detecting localised shape similarity, and wish to be invariant to scale and offset. Although there is no reference to normalisation in [19, 20], an amortised constant time normalisation method based on maintaining updated statistics is proposed in [13]. We have separated the procedures *findDistances* and *assessCandidate* for clarity, but for the implementation they may be combined to take advantage of speed-up techniques.

2.2.2 *Measuring Distances*

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

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

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

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

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

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

Note that since $d_{i,S}$ is a minima, [20] propose speeding up the calculation of $d_{i,S}$ with an early abandon.

2.2.3 *Measuring The Quality of a Shapelet*

The original shapelet papers use information gain to determine the quality of a shapelet [19, 20, 13]. This involves sorting the distance list D_S, then evaluating the information gain on the class values for each possible split value. [20] propose an early abandon on calculating D_S through maintaining an upper bound on the *quality* of the candidate whilst generating D_S. If this upper bound falls below the *best* found so far, the calculation of D_S can be abandoned. After each $d_{i,S}$ is found, the current best information gain split is calculated and the upper bound is found by assuming the most optimistic division of the remaining distances.

3. SHAPELET TRANSFORM

The transformation we propose handles shapelets in three distinct stages, Firstly, the algorithm conducts a single scan of the data to extract the best k shapelets. Note that whilst k is a parameter to set, it is simply a cut-off value for the maximum number of shapelets to store and has no effect on the quality of the individual shapelets that are extracted. Secondly, an appropriate value for the number of shapelets to use in the final transformed data set is estimated using a simple cross-validation approach. Finally, a new transformed data set is created where each attribute represents a shapelet, and the value of each attribute is the distance from the shapelet to the original series. The key motivation for transforming the data in this way is that we can disassociate shapelet finding from building a classifier, allowing the transformed data set to be used in conjunction with any classifier.

3.1 Alternative Quality Measure

As discussed in Section 2.2, the original shapelet decision tree uses information gain to assess the quality of a candidate. The motivation for this is two-fold; firstly, information gain is suitable for identifying how to produce a partition of the data, which is essential when it is necessary to recursively divide the data. Secondly, it is possible to use information gain in conjunction with the upper bounding/early abandon technique described in [20]. We use an alternative to information gain for the following reasons.

Firstly, since we are trying to generate a set of shapelets from the entire data set, our concern is not necessarily how well a candidate splits the data. Rather, we are concerned with how the distribution of the distances of the alternative classes differ. To elaborate, if we have the list of distances D_S, rather than find the best way of partitioning D_S we address the issue of how different are the lists $D_S^1, D_S^2, \ldots, D_S^c$, where D_S^j contains all the distances from the candidate to time series of class j.

Secondly, the upper bounding technique for information gain relies on identifying the ideal partition of a number of unevaluated distances. However, the utility of this approach degrades with multi-class problems, as a simple binary split is impossible with 3 or more class values. In the most pessimistic cases, all possible optimistic combinations of unevaluated distances must be considered and can quickly become untenable.

There are several alternative approaches we could adopt to assess the difference in distributions between the class distances. The simplest approach, which we adopt, is to use the F-statistic used for the difference of means in an ANOVA. Whilst a non-parametric test, such as multiple-sample Mann-Whitney, would possibly be more robust, the fact that we are not actually performing a hypothesis test (but instead using the test statistic as a comparative metric) means the reduced power of the F-statistic in the face of outliers is less of a problem. Hence the *assessCandidate* method we use is simply the F-statistic of a fixed effects ANOVA.

3.2 Shapelet Generation

The process of extracting the k best shapelets is defined in Algorithm 2.

Algorithm 2 ShapeletCachedSelection(T, min, max, k)

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

The algorithm begins by processing the data in a very similar manner to the original shapelet selection algorithm of [20], defined earlier in Algorithm 1. For each series in the data set, all subsequences of all possible lengths according to the *min* and *max* length parameters are visited. However, unlike Algorithm 1, where all candidates are assessed and only the best is stored, the caching algorithm stores all candidates for a given time series along with their associated quality measures (line 10). Once all candidates of a series have been assessed, the candidates are sorted in order of fitness and self-similar shapelets are removed. We define two shapelets as being self-similar if they are taken from the same series and have any overlapping indices. Once we have the set of non self-similar shapelets for a series, we merge these with the current best shapelets and retain the top k, and continue to iterate through the data until all series have been processed. Note that we do not store all candidates indefinitely; after processing each series, we simply retain those that belong to the best k so far overall, and discard all other shapelets. This means that we can avoid the large space overhead that would be caused by retaining all possible candidates.

We choose to store shapelets from a series separately and merge them later, rather than keeping the best k on the fly, to ensure that we are left with the best shapelets after removing self-similar candidates. For example, if we have a set of the best k shapelets at a given point in the series, the next candidate that is processed could potentially overlap two or perhaps more shapelets in the store. If this current shapelet is deemed to be better, all overlapping self-similar candidates must be removed from the store. A problem would then arise if a better shapelet is found later in the series that overlaps the one currently being stored, but isn't self-similar to at least one of the previously removed candidates. As each subsequence is only ever visited once, it would be impossible to get back the previously removed candidates, even if they ultimately should be amongst the best k shapelets. Therefore to avoid this problem, we initially extract all possible shapelets from a series, sort them by their quality and then remove those that are self similar in order. This gives priority to the best shapelets and ensures that no candidates are prematurely removed. We can then safely merge these with the existing k best shapelets before moving on to processes the next series. At this point it is safe to discard all other shapelets from the current series as they are not in the top k and cannot be self-similar to shapelets from another series.

3.2.1 Length Parameter Approximation

Similarly to the original shapelet finding algorithm of [20], the algorithm that we define in Algorithm 2 requires two length parameters to be set: min and max. These values define the range of possible candidate shapelet lengths, which provides scope to the search and can help make the algorithm more efficient. However, setting the parameters incorrectly can be detrimental to the outcome of the shapelet transformation if they prevent the most informative subsequences from being considered. To accommodate running the shapelet filter on a range of data sets without any specialised knowledge of the data, we define a simple algorithm for estimating the min and max parameters.

Algorithm 3 EstimateMinAndMax(T)

```
shapelets = ∅;
length = $T_1.length$;
for $i = 1$ to 10 do
    randomiseOrder($T$);
    $T' = [T_1, T_2, ..., T_{10}]$;
    currentShapelets = ShapeletCachedSelection($T'$, 1, length, 10);
    shapelets.add(currentShapelets);
end for
orderByLength(shapelets);
min = $shapelets_{25}.length$;
max = $shapelets_{75}.length$;
return min, max;
```

The procedure outlined in Algorithm 3 takes 10 random series from the dataset T and uses Algorithm 2 to find the 10 best shapelets within this small subset of data. The search parameters here are set from 1 to the length of a whole series in T, meaning no constraints are placed on the length of shapelets in this search. This is repeated 10 times in total, producing a set of 100 shapelets. The shapelets are sorted in order of their length and the lengths of the 25th and 75th shapelets are extracted and returned as min and max respectively. Note that this will not necessarily result in the optimal solution for parameter finding. However, it was important that we could adopt an automatic approach to approximate min and max parameters across a number of datasets to allow us to compare our filter fairly against the original tree implementation of shapelets. Therefore we use this approach to approximate min and max for each data set and build all shapelet trees and filters using these values.

3.3 Data Transformation

One of the main motivations of the proposed transformation is to allow shapelets to be used with a diverse range of classification algorithms. Rather than restricting them to classification through decision tree structures, our algorithm uses shapelets to transform instances of data into a number of features that can then be treated as a generic classification problem. The transformation process is defined in Algorithm 4.

Algorithm 4 FilterData(Shapelets S, Dataset D)

```
output = ∅;
for all time series $ts$ in $D$ do
    transformed = ∅;
    for all shapelets $s$ in $S$ do
        dist = subsequenceDist($ts$, $s$);
        transformed.add(dist);
    end for
    output.add(transformed);
end for
return output;
```

The transformation process is carried out using the subsequence distance calculation described in Section 2.2.2. Firstly, a set of k shapelets, S, is generated from the training data T, as seen in the previous section. For each instance of data T_i, the subsequence distance is computed between T_i and S_j, where $j = 1, 2, \ldots, k$. The resulting k distances are used to form a new instance of transformed data, where each attribute corresponds to the distance from each shapelet to the original time series. When using data split into training and test partitions, the shapelet extraction is carried out on only the training data to avoid bias; these shapelets are then used to transform each instance of the training and test data to create transformed data sets, which can then be used with any traditional classification algorithm.

3.4 Shapelet Selection

Using k shapelets in the filter will not necessarily yield the best data for classification. Using too few shapelets would not provide enough information to make informed classification decisions, whilst using too many could overfit classifiers trained with the transformed data and dilute the influence of important shapelets. In the experiments contained in this paper, we use two strategies for selecting the number of shapelets to use in the filter for a given set of data; firstly, as a benchmark we use $\frac{n}{2}$ shapelets in the filter, where n is the number of readings in a single series of the data. The second approach automatically selects the number of shapelets to use based on the results of a 5-fold cross-validation experiment.

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

The value of n with the best overall accuracy across all five folds of the data is selected as the number of shapelets to use in the final filter. In cases where multiple values obtain the best results, we evaluated three strategies for selecting a single value from the set of best values: pick the smallest, middle or largest value. We found that picking the maximum marginally outperformed using the middle value, whilst both approaches performed better than selecting the smallest value. For brevity, we do not include the experiments for this here, but they can be found at [15].

4. EXPERIMENTAL DESIGN

To evaluate the shapelet transform, we selected 18 data sets from the UCR time series repository (listed in Table 1) and 8 new data sets provided by us. We selected these particular data sets because they have relatively few cases; even with optimisation, the shapelet algorithm is time consuming. At the time of writing, we have not evaluated the transform on any other data sets. We use a simple train/test split and all reported results are testing accuracy. All shapelet selection, model selection, and classifier construction is done exclusively on the training set, whilst the test set is only used once with the final trained classifier. All algorithms and experiments were implemented in Java within the Weka framework, and the shapelet transform is implemented as a Weka batch filter to allow easy integration into existing classification code. All the code to generate our results is available at [15].

4.1 New Data Sets

We provide 8 new data sets that contain data taken from hand x-rays, originally from [9]. The data consists of images focusing on eight specific bones of the hand, where each data set contains 1045 cases of a single type of bone. The eight bones in the data are the proximal phalanges of the thumb, little and middle fingers; the middle phalanges of the little and middle fingers; and the distal phalanges of the thumb, middle and little fingers. Each instance of data has a class label that is either infant (0-6 years), junior (7-12 years) or teen (13-18) years. This approach is similar to the system proposed in [17].

For each of the 8 bones, the data is partitioned into 200 instances of training and 845 instances of test data. The images were converted to 1-d series by initially calculating the outlines of the 1045 hand x-ray images using the algorithm described in [4]. The locations of the tips and webs of the hand were extracted from these outlines using the algorithm presented in [3]. From these positions, the axes of the thumb, middle and little fingers were calculated by finding the midpoint between adjacent webs in the hand. In the cases of the thumb and little finger where there is only one adjacent web for each, the axes were approximated by extending the line from the previous finger and calculating where this intersects the hand outline. Once provided with these axes, region-of-interest boxes were calculated for each of the eight bones (as shown in Figure 1). Each box was warped into a rectanglular-shaped mesh (500 × 150 pixels) using a piecewise affine warp to create a new image for each of the 8 bones. Each image was converted into 1-d series, creating 1045 instances of data for each of the 8 bone types.

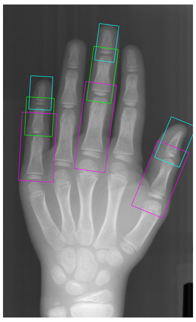

Figure 1: A hand x-ray with the eight bones boxed: proximal (purple, bottom), middle (green, middle), and distal (blue, top) phalanges. Note, the thumb does not have a middle phalange.

A common method for converting an image into a 1-d series is to calculate a histogram of the image [14]. However, this approach doesn't incorporate any location information from the original image. To preserve this contextual information, we convert images to 1-d series by resizing them to 30 × 9 pixels, which is represented as a vector of length 270. This allows us to retain location information in the data, whilst converting the images into 1-d series that can be posed as a TSC problem.

5. RESULTS

We have proposed several changes to the way shapelets can be used for classification and present a range of experiments to test these changes.

5.1 Shapelets: Embedded vs. Transformed

Our first objective is to establish that separating shapelets from the classifier does not reduce classification accuracy. We implemented a shapelet decision tree classifier as described in [20], and compared results to a C4.5 decision tree trained and tested on shapelet transformed data (using information gain as the quality measure and $\frac{n}{2}$ shapelets). Table 1 shows the results for the 26 data sets used. The Shapelet tree was best on 13, C4.5 best on 12 and they were tied on one. There is no significant difference detected by a paired t-test or a Wilcoxon signed rank test. There is no evidence that performing the shapelet extraction prior to constructing the decision tree makes the classifier less accurate.

Table 1: Shapelet tree classification vs. C4.5 classification with $\frac{n}{2}$ shapelet filtered features

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

5.2 Information Gain vs F-Statistic

Our second experiment is designed to compare the F-statistic to information gain for shapelet selection. Figure 2 compares the accuracy of a C4.5 classifier built on a shapelet transform using information gain and the F-statistic in the style presented in [7]. Information gain slightly outperforms the F-statistic, winning 15 of the 26 comparisons. Whilst this single experiment is not significant, further experiments with other classifiers indicated that this pattern was repeated. Qualitatively at least, we think information gain may produce more discriminatory shapelet features. Table 2 compares the build time for the shapelet tree, the information gain shapelet transform and the F-statistic shapelet transform. The F-statistic filter is faster on all but one of the data sets. However, there is an important caveat to these results; our shapelet tree implementation does not employ the information gain upper bound. We found that for problems with multiple classes, it was somewhat counter productive, with the requirement to recalculate split points offsetting any early abandon gains. However, we recognize this may be due to our implementation, and acknowledge a potential source of implementation bias. We conclude that whilst there is a case for using the F-statistic over information gain on performance grounds, particularly on large multi-class problems, the marginal accuracy differential means that we continue with information gain.

5.3 Shapelet Transformation Classifiers

The main contribution of this paper is to demonstrate how moving the shapelet discovery outside of the classification algorithm can improve the overall accuracy. Table 3 shows the classification test accuracy of the shapelet tree, C4.5, 1-NN, Naive Bayes, a Bayesian Network, Random Forest,

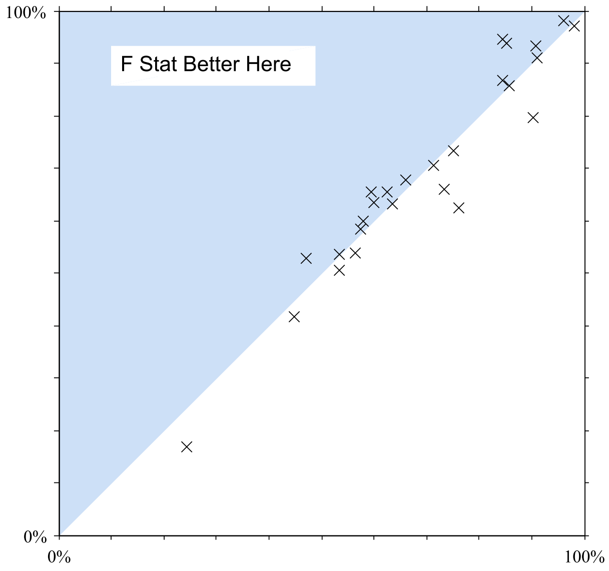

Figure 2: A comparison of using the F-stat quality criteria against information gain for the shapelet transform. Accuracy results were obtained with a C4.5 classifier built on the transformed data set.

Table 2: Build times of shapelet tree, information gain shapelet transform and F-stat transform (time in seconds)

Data	Shapelet Tree	Info Gain Transform	F-Stat Transform
Adiac	31403.17	20896.47	**5321.30**
Beef	2501.50	1492.31	**1458.21**
ChlorineConcentration	121019.15	31723.73	**18595.21**
Coffee	305.17	**295.07**	298.51
DiatomSizeReduction	100.88	68.65	**66.27**
ECGFiveDays	189.52	192.56	**187.30**
ElectricDevices	5524.90	3040.93	**2187.11**
FaceFour	8653.46	5821.65	**5757.35**
GunPoint	743.69	731.84	**708.59**
ItalyPowerDemand	5.09	4.17	**1.94**
Lighting7	35877.14	20279.77	**19748.46**
MedicalImages	36227.04	19877.87	**9693.09**
MoteStrain	11.13	11.39	**10.22**
SonyAIBORobotSurface	9.36	9.61	**8.08**
Symbols	17539.63	10833.19	**10790.84**
SyntheticControl	4312.67	2801.06	**1180.79**
Trace	108658.88	72401.39	**71566.33**
TwoLeadECG	5.35	5.41	**4.46**
DP_Little	13447.14	5941.87	**4854.11**
DP_Middle	16426.07	8414.13	**7261.95**
DP_Thumb	24074.84	12798.50	**10868.34**
MP_Little	33585.39	19364.60	**17456.99**
MP_Middle	20049.11	8370.00	**7176.03**
PP_Little	44027.99	23555.03	**20518.56**
PP_Middle	32190.56	15115.02	**13513.17**
PP_Thumb	22505.93	11749.81	**10130.71**

Table 3: Testing accuracy and ranks of 8 classifiers constructed on the shapelet transform with $n/2$ shapelets.

Data	Shapelet Tree		C4.5		1NN		Naïve Bayes		Bayesian Network		Random Forest		Rotation Forest		SVM (linear)	
Adiac	29.92%	(3)	24.30%	(7)	25.32%	(5)	28.13%	(4)	25.06%	(6)	30.43%	(2)	**30.69%**	**(1)**	23.79%	(8)
Beef	50.00%	(8)	60.00%	(6.5)	83.33%	(3)	73.33%	(4)	**90.00%**	**(1)**	60.00%	(6.5)	70.00%	(5)	86.67%	(2)
ChlorineConcentration	58.80%	(2)	56.48%	(6)	56.93%	(5)	45.96%	(8)	57.08%	(4)	57.58%	(3)	**63.52%**	**(1)**	56.15%	(7)
Coffee	96.43%	(4.5)	85.71%	(8)	**100.00%**	**(1.3)**	92.86%	(6)	96.43%	(4.5)	**100.00%**	**(1.3)**	89.29%	(7)	**100.00%**	**(1.3)**
DiatomSizeReduction	72.22%	(8	75.16%	(7)	**93.46%**	**(1)**	78.76%	(6)	90.20%	(3)	80.39%	(5)	83.01%	(4)	92.16%	(2)
ECGFiveDays	77.47%	(8)	96.17%	(6)	98.37%	(4)	96.40%	(5)	**99.54%**	**(1)**	93.26%	(7)	98.61%	(3)	98.95%	(2)
ElectricDevices	54.90%	(2)	53.45%	(4)	24.25%	(7)	25.37%	(5)	53.63%	(3)	**55.98%**	**(1)**	24.25%	(7)	24.25%	(7)
FaceFour	84.09%	(7)	76.14%	(8)	**100.00%**	**(1.5)**	97.73%	(4.5)	**100.00%**	**(1.5)**	87.50%	(6)	98.86%	(3)	97.73%	(4.5)
GunPoint	89.33%	(8)	90.67%	(7)	98.00%	(4)	92.00%	(6)	99.33%	(2)	96.00%	(5)	98.67%	(3)	**100.00%**	**(1)**
ItalyPowerDemand	89.21%	(8)	90.96%	(7)	92.13%	(4.5)	92.52%	(2)	92.42%	(3)	**93.00%**	**(1)**	92.03%	(6)	92.13%	(4.5)
Lighting7	49.32%	(7.5)	53.42%	(6)	49.32%	(7.5)	57.53%	(5)	65.75%	(2.5)	64.38%	(4)	65.75%	(2.5)	**69.86%**	**(1)**
MedicalImages	48.82%	(4)	44.87%	(6)	45.66%	(5)	17.37%	(8)	28.16%	(7)	50.79%	(3)	51.45%	(2)	**52.50%**	**(1)**
MoteStrain	82.51%	(8)	84.42%	(7)	**90.34%**	**(1)**	88.82%	(3)	89.06%	(2)	84.58%	(6)	86.98%	(5)	88.66%	(4)
SonyAIBORobotSurface	84.53%	(6)	84.53%	(5)	84.03%	(7)	79.03%	(8)	**89.68%**	**(1)**	85.19%	(4)	89.02%	(2)	86.69%	(3)
Symbols	77.99%	(6)	47.14%	(8)	85.63%	(2)	77.99%	(7)	**92.26%**	**(1)**	84.62%	(3.5)	84.42%	(5)	84.62%	(3.5)
SyntheticControl	**94.33%**	**(1)**	90.33%	(4)	93.00%	(2)	78.00%	(7)	76.67%	(8)	89.00%	(5)	92.00%	(3)	87.33%	(6)
Trace	98.00%	(5)	98.00%	(5)	98.00%	(5)	98.00%	(5)	**100.00%**	**(1)**	98.00%	(5)	98.00%	(5)	98.00%	(5)
TwoLeadECG	85.07%	(8)	85.25%	(7)	**99.47%**	**(1)**	99.12%	(3)	98.77%	(4)	96.14%	(6)	97.98%	(5)	99.30%	(2)
DP_Little	65.44%	(8)	65.92%	(7)	72.78%	(6)	73.49%	(3)	72.90%	(5)	73.02%	(4)	74.67%	(2)	**75.15%**	**(1)**
DP_Middle	70.53%	(8)	71.24%	(7)	73.73%	(6)	73.96%	(5)	74.67%	(4)	75.50%	(3)	76.80%	(2)	**79.64%**	**(1)**
DP_Thumb	58.11%	(7)	57.99%	(8)	60.71%	(6)	62.96%	(5)	63.91%	(4)	64.14%	(3)	67.10%	(2)	**69.82%**	**(1)**
MP_Little	66.39%	(7)	63.43%	(8)	68.52%	(6)	68.76%	(5)	69.47%	(4)	71.36%	(3)	**75.15%**	**(1)**	75.03%	(2)
MP_Middle	71.01%	(7)	73.25%	(4)	70.89%	(8)	71.95%	(5)	71.12%	(6)	75.15%	(2)	74.67%	(3)	**76.92%**	**(1)**
PP_Little	59.64%	(7)	57.40%	(8)	67.22%	(5)	69.23%	(4)	70.06%	(2)	66.63%	(6)	69.82%	(3)	**72.07%**	**(1)**
PP_Middle	61.42%	(8)	62.49%	(7)	68.52%	(6)	69.82%	(5)	71.36%	(3)	70.53%	(4)	75.38%	(2)	**75.86%**	**(1)**
PP_Thumb	60.83%	(7)	59.53%	(8)	67.69%	(6)	69.35%	(4)	69.47%	(3)	67.81%	(5)	72.78%	(2)	**75.50%**	**(1)**
Average Rank	6.27		6.60		4.48		5.10		3.33		4.04		3.33		2.8	

Rotation Forest and a Support Vector Machine, all with default Weka settings. The support vector machine is the best performing classifier, with an average rank of 2.86 and best performance in 10 out of 26 problems. Figure 3 shows a critical difference diagram as described in [5]. This diagram is derived from the overall test of significance in mean ranks, and groups classifiers together into *cliques*, illustrated by the bars. There is a clear division in performance between the simpler classifiers (shapelet tree, C4.5, Naive Bayes and 1-NN) and the more complex classifiers (Rotation forest, Random Forest, Bayesian Networks and SVM). Whilst there may be a trade off between interpretability and accuracy (a tree is easier to understand than a SVM), our point is that by separating the shapelet discovery, there is greater potential to explore possible solutions.

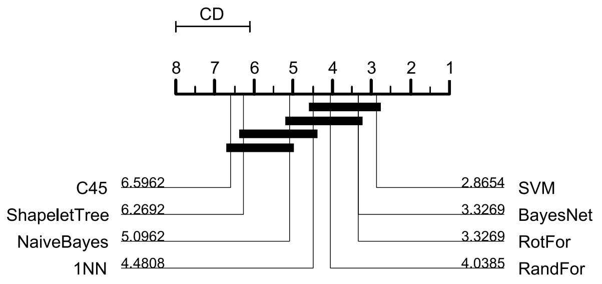

Figure 3: Critical difference plot for eight shapelet based classifiers derived from the results in Table 3

5.4 Shapelets Selection

In Section 3.4, we defined a method for shapelet selection through cross validation. To demonstrate the utility of this technique, we calculated the number of shapelets to use for each data set and repeated the classification experiments from Table 3. The results presented in Table 4 show the relative performance change of these classifiers against simply training the classifiers with $n/2$ shapelets, as in Table 3. The results show that using an automatically selected number of shapelets set through cross-validation improves the average classification accuracy of 7 out of the 8 classifiers. In all cases, the classifiers achieve equal or better results on over half of the data sets, with the Random Forest classifier improving the most overall.

Table 5: Accuracy of the SVM built on a shapelet transform, Dynamic Time Warping with 1-NN on the raw data and an ensemble of 1-NN classifiers built on alternative transformations (see [1])

	Shapelet SVM	DTW 1-NN	Ensemble
Adiac	31.20%	61.13%	64.45%
Beef	83.33%	63.33%	60.00%
Coffee	100.00%	92.86%	82.14%
ElectricDevices	24.25%	65.00%	62.21%
FaceFour	98.86%	81.82%	86.36%
GunPoint	100.00%	91.33%	95.33%
Lighting7	71.23%	71.23%	69.86%
SyntheticControl	90.67%	97.67%	91.00%
Trace	98.00%	99.00%	81.00%

5.5 Other Classifiers

Our final accuracy results are used to demonstrate that there are problems where a shapelet approach will outperform others. Table 5 shows the accuracy of three classifiers trained with 9 of the 26 datasets. The classifiers are: an SVM built on selected shapelets (results comparable with Table 4), Dynamic Time Warping with 1-NN on the raw data and an ensemble of 1-NN classifiers built on transformations into the power spectrum, autocorrelation function and principle component space (described in [1]). These data sets were selected as they are common to both papers.

The comparison is provided for information only; we have obviously introduced some selection bias by choosing the best classifier (SVM) from out experiments. Nevertheless, it is interesting to note that the shapelet approach is clearly the best on some data sets (Beef, Coffee and FaceFour), and yet fails dramatically on others (Adiac and ElectricDevices). This offers promising support for shapelet-based approaches, suggesting that they fill a classification niche that has not been covered in the literature.

Table 4: Relative accuracies of the classifiers when trained with an automatically selected number of shapelets through cross-validation vs. using $n/2$ shapelets (as in Table 3)

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

5.6 Exploratory Data Analysis

The results we have reported show that using shapelets as a filter can provide some very promising classification results. However, one of the strengths of using shapelets as a classification tool is that they allow a level of interpretation that other classification approaches simply do not. The original work on shapelets in [19] demonstrated this, with a number of examples where they show the decision trees that they train on datasets and the associated shapelets.

One of the key motivations behind our work using shapelets for TSC is to produce accurate and reliable classification decisions that are interpretable. To demonstrate that our filter retains this desirable trait of the original shapelet implementation, we briefly analyse one of our previous experiments using the GunPoint data set. The GunPoint data contains time series of an actor carrying out the motion of drawing a gun, and the classification problem is to determine whether or not they were holding a prop or just miming the action (the *Gun/NoGun* problem). In [19], they identified that the most important shapelet for classification was when the actor lowered their arm; if they had no gun, a phenomenon called overshoot occurred and caused a dip in the data. This is summarised in Figure 4, originally presented in [19].

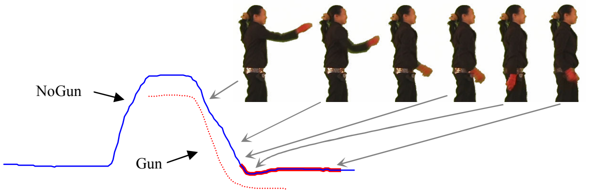

Figure 4: An illustration of the *Gun/NoGun* problem taken from [19]. The shapelet that they extract is highlighted at the end of the series.

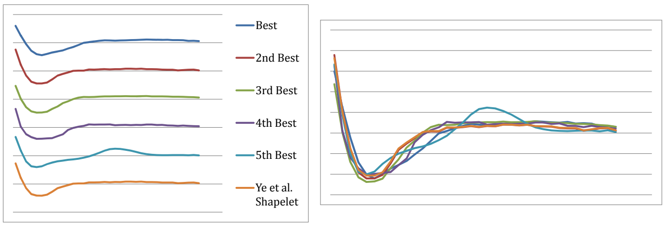

Figure 5: An illustration of the five best shapelets extracted by our filter. The graph to the right shows how closely matched they are when place on top of one another.

The shapelet decision tree trained by [19] contains a single shapelet corresponding to the arm being lowered back into position at the end of the series. To demonstrate that our filter agrees with this and extracts the important information from the data, we filtered the GunPoint data set using the length parameters specified in the original paper to allow for a fair comparison between the two methods. The top five shapelets that we extracted are presented in Figure 5, along with the shapelet reported by [19].

The graphs in Figure 5 show that each of the top five shapelets from our filter were very closely matched with the shapelet from [19], reinforcing the notion that our filter produces interpretable results. Furthermore, if we extract the top ten shapelets from the filter we can gain even further insight. Figure 6 shows that the top ten shapelets form two distinct clusters. Interestingly, the shapelets to the left of the figure correspond to the moments where the arm is lifted and are instances where there is a gun. These shapelets could correspond to the subtle extra movements required to lift the prop, aiding classification by providing more information.

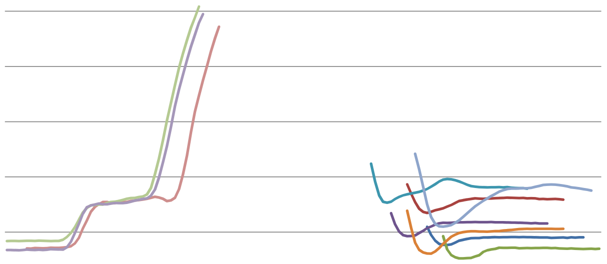

Figure 6: The 10 best shapelets for the *Gun/NoGun* problem as extracted by our filter. The shapelets form two distinct clusters, the first where the arm is raised and the second when the arm is lowered.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a shapelet transform for TSC that extracts the k best shapelets from a dataset in a single pass. We implement this using a novel caching algorithm to store shapelets, and apply a simple, parameter-free cross-validation approach for extracting the most significant shapelets. We transform a total of 26 data sets with our filter and demonstrate that a C4.5 decision tree classifier trained with transformed data is competitive with an implementation of the original shapelet decision tree of [19]. We show that our filtered data can be applied to further, non-tree based classifiers to achieve improved classification performance, whilst still maintaining the interpretability of shapelets. We provide two implementations of the filter using different quality measures for discriminating between shapelets; we use information gain as proposed by [19] in the first, and introduce the application of the F-statistic as an evaluation method for shapelets in the second. We show that classifiers trained using features derived from an F-statistic filter are competitive with classifiers trained with the information gain approach, whilst being easier to apply to multi-class classification problems. Finally, we provide exploratory data analysis of the shapelets extracted by our filter on the *Gun/NoGun* problem and compare them with the output of [20]. We show that the shapelets we find are consistent with the discriminatory shapelet in the original work, and show that our approach can lead to further insight into the problem by looking at a number of the top shapelets.

Future direction of our work could involve investigating how we handle extracted shapelets. We have shown that using a number of shapelets to transform data can lead to strong classification results, but it is obvious that some shapelets produced across a data set may be very similar. An interesting extension of our work would be to perform a cluster analysis on the shapelets produced by our filter, and then train classifiers using data transformed by single shapelets of distinct clusters, rather than the top k shapelets.

7. REFERENCES

[1] A. Bagnall, L. Davis, J. Hills, and J. Lines. Transformation based ensembles for time series classification. In *Proc. 12th SDM*, 2012.

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

[3] L. Davis, B. Theobald, J. Lines, A. Toms, and A. Bagnall. On the segmentation and classification of hand outlines (under review). *International Journal of Neural System*, 2012.

[4] L. Davis, B. J. Theobald, A. Toms, and A. Bagnall. On the extraction and classification of hand outlines. In *Proc. of the 12th IDEAL*, 2011.

[5] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7:1–30, 2006.

[6] H. Deng, G. Runger, E. Tuv, and M. Vladimir. A time series forest for classification and feature extraction. Technical report, Arizona State University, 2011.

[7] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. In *Proc. 34th VLDB*, 2008.

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

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

[10] Y. Jeong, M. Jeong, and O. Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44:2231–2240, 2010.

[11] 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.

[12] F. Mörchen, I. Mierswa, and A. Ultsch. Understandable models of music collections based on exhaustive feature generation with temporal statistics. In *12th International Conference on Knowledge Discovery in Data and Data Mining (ACM SIGKDD 2006)*, pages 882–891, 2006.

[13] A. Mueen, E. Keogh, and N. Young. Logical-shapelets: An expressive primitive for time series classification. In *Proc. 17th ACM SIGKDD*, 2011.

[14] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and M. Westover. Exact discovery of time series motifs. In *Proc. 9th SDM*, 2009.

[15] paper authors. Accompanying information for this paper. `https://sites.google.com/site/shapelettransform/`.

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

[17] H. Thodberg, S. Kreiborg, A. Juul, and K. Pedersen. The bonexpert method for automated determination of skeletal maturity. *IEEE Trans. Med. Imaging*, 28(1):52–66, 2009.

[18] Z. Xing, J. Pei, P. Yu, and K. Wang. Extracting interpretable features for early classification on time series. In *Proc. 11th SDM*, 2011.

[19] L. Ye and E. Keogh. Time series shapelets: A new primitive for data mining. In *Proc. 15th ACM SIGKDD*, 2009.

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