

Binary Shapelet Transform for Multiclass Time Series Classification

Aaron Bostrom and Anthony Bagnall

University of East Anglia, Norwich, UK, NR47TJ.

Abstract. Shapelets have recently been proposed as a new primitive for time series classification. Shapelets are subseries of series that best split the data into its classes. In the original research, shapelets were found recursively within a decision tree through enumeration of the search space. Subsequent research indicated that using shapelets as the basis for transforming datasets leads to more accurate classifiers.

Both these approaches evaluate how well a shapelet splits all the classes. However, often a shapelet is most useful in distinguishing between members of the class of the series it was drawn from against all others. To assess this conjecture, we evaluate a one vs all encoding scheme. This technique simplifies the quality assessment calculations, speeds up the execution through facilitating more frequent early abandon and increases accuracy for multi-class problems. We also propose an alternative shapelet evaluation scheme which we demonstrate significantly speeds up the full search.

1 Introduction

Time series classification (TSC) is a subset of the general classification problem, the primary difference being that the ordering of attributes within each instance is important. For a set of n time series, $\mathbf{T} = \{T_1, T_2, \dots, T_n\}$, each time series has m ordered real-valued observations $T_i = \{t_{i1}, t_{i2}, \dots, t_{im}\}$ and a class value c_i . The aim of TSC is to determine a function that relates the set of time series to the class values.

One recently proposed technique for TSC is to use shapelets [17]. Shapelets are subseries of the series \mathbf{T} that best split the data into its classes. Shapelets can be used to detect discriminatory phase independent features that cannot be found with whole series measures such as dynamic time warping. Shapelet based classification involves measuring the similarity between a shapelet and each series, then using this similarity as a discriminatory feature for classification. The original shapelet-based classifier [17] embeds the shapelet discovery algorithm in a decision tree, and uses information gain to assess the quality of candidates. A shapelet is found at each node of the tree through an enumerative search. More recently, we proposed using shapelets as a transformation [9]. The shapelet transform involves a single-scan algorithm that finds the best k shapelets in a set of n time series. We use this algorithm to produce a transformed dataset, where each of the k features is the distance between the series and one shapelet.

Hence, the value of the i^{th} attribute of the j^{th} record is the distance between the j^{th} record and the i^{th} shapelet. The primary advantages of this approach are that we can use the transformed data in conjunction with any classifier, and that we do not have to search sequentially for shapelets at each node. However, it still requires an enumerative search throughout the space of possible shapelets and the full search is $O(n^2m^4)$. Improvements for the full search technique were proposed in [17, 12] and heuristics techniques to find approximations of the full search were described in [15, 6, 4].

Our focus is only on improving the exhaustive search. One of the problems of the shapelet search is the quality measures assess how well the shapelet splits all the classes. For multi-class problems measuring how well a shapelet splits all the classes may confound the fact that it actually represents a single class. Consider, for example, a shapelet in a data set of heartbeat measurements of patients with a range of medical conditions. It is more intuitive to imagine that a shapelet might represent a particular condition such as arrhythmia rather than discriminating between multiple conditions equally well. We redefine the transformation so that we find shapelets assessed on their ability to distinguish one class from all, rather than measures that separate all classes. This improves accuracy on multi-class problems and allows us to take greater advantage of the early abandon described in [17].

A further problem with the shapelet transform is that it may pick an excessive number of shapelets representing a single class. By definition, a good shapelet will appear in many series. The best way we have found to deal with this is to generate a large number of shapelets then cluster them [9]. However, there is still a risk that one class is generally easier to classify and hence has a disproportionate number of shapelets in the transform. The binary shapelet allows us to overcome this problem by balancing the number of shapelets we find for each class.

Finally, we describe an alternative way of enumerating the shapelet search that facilitates greater frequency of early abandon of the distance calculation.

2 Shapelet Based Classification

The shapelet transform algorithm described in [9] is summarised in Algorithm 2. Initially, for each time series, all candidates of length *min* to *max* are generated (i.e. extracted and normalised in the method *generateCandidates*). Then the distance between each shapelet and the other $n - 1$ series are calculated to form the order list D_S . Distance between a shapelet S length l and a series T is given by

$$sDist(S, T) = \min_{w \in W_l} (dist(S, w)) \quad (1)$$

where W_l is the set of all l length subseries in T and *dist* is the Euclidean distance between the equal length series S and w . The order list is used to determine the quality of the shapelet in the *assessCandidate* method. Quality can be assessed by information gain [17] or alternative measures such as the F, modes median or rank order statistic [11]. Once all the shapelets for a series are evaluated they are

sorted and the lowest quality overlapping shapelets are removed. The remaining candidates are then added to the shapelet set. By default, we set $k = 10n$ with the caveat that we do not accept shapelets that have zero information gain.

Algorithm 1 FullShapeletSelection(\mathbf{T} , min , max , k)

Input: A list of time series \mathbf{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:  $kShapelets \leftarrow \emptyset$ 
2: for all  $T_i$  in  $\mathbf{T}$  do
3:    $shapelets \leftarrow \emptyset$ 
4:   for  $l \leftarrow min$  to  $max$  do
5:      $W_{i,l} \leftarrow generateCandidates(T_i, l)$ 
6:     for all subseries  $S$  in  $W_{i,l}$  do
7:        $D_S \leftarrow findDistances(S, \mathbf{T})$ 
8:        $quality \leftarrow assessCandidate(S, D_S)$ 
9:        $shapelets.add(S, quality)$ 
10:     $sortByQuality(shapelets)$ 
11:     $removeSelfSimilar(shapelets)$ 
12:     $kShapelets \leftarrow merge(k, kShapelets, shapelets)$ 
13: return  $kShapelets$ 

```

Once the best k shapelets have been found, the transform is performed with Algorithm 2. A more detailed description can be found in [8].

Extensions to the basic shapelet finding algorithm can be categorised into techniques to speed up the average case complexity of the exact technique and those that use heuristic search. The approximate techniques include reducing the dimensionality of the candidates and using a hash table to filter [15], searching the space of shapelet values (rather than taking the values from the train set series) [6] and randomly sampling the candidate shapelets [4]. Our focus is on improving the accuracy and speed of the full search. Two forms of early abandon described in [17] can improve the average case complexity. Firstly, the Euclidean distance calculations within the $sDist$ (Equation 1) can be terminated early if they exceed the best found so far. Secondly, the shapelet evaluation can be abandoned early if $assessCandidate$ is updated as the $sDist$ are found and the best possible outcome for the candidate is worse than the current top candidates.

A speedup method involving trading memory for speed is proposed in [12]. For each pair of series T_i, T_j , cumulative sum, squared sum, and cross products of T_i and T_j are precalculated. With these statistics, the distance between subseries can be calculated in constant time, making the shapelet-discovery algorithm $O(n^2m^3)$. However, precalculating of the cross products between all series prior to shapelet discovery requires $O(n^2m^2)$ memory, which is infeasible for most problems. Instead, [12] propose calculating these statistics prior to the start of the scan of each series, reducing the requirement to $O(nm^2)$ memory, but increasing the time overhead. Further refinements applicable to shapelets were

described in [14], most relevant of which was a reordering of the sequence of calculations within the *dist* function to increase the likelihood of early abandon. The key observation is that because all series are normalised, the largest absolute values in the candidate series are more likely to contribute large values in the distance function. Hence, if the distances between positions with larger candidate values are evaluated first, then it is more likely the distance can be abandoned early. This can be easily implemented by creating an enumeration through the normalised candidate at the beginning, and adds very little overhead. We use this technique in all experiments.

Algorithm 2 *FullShapeletTransform*(Shapelets \mathbf{S}, \mathbf{T})

```

1:  $\mathbf{T}' \leftarrow \emptyset$ 
2: for all  $T$  in  $\mathbf{T}$  do
3:    $T' \leftarrow \langle \rangle$ 
4:   for all shapelets  $S$  in  $\mathbf{S}$  do
5:      $dist \leftarrow sDist(S, T)$ 
6:      $T' \leftarrow append(T', dist)$ 
7:    $T' \leftarrow append(T', T.class)$ 
8:  $\mathbf{T}' \leftarrow \mathbf{T}' \cup T'$ 
9: return  $\mathbf{T}'$ 

```

2.1 Classification technique

Once the transform is complete we can use any classifier on the problem. To reduce classifier induced variance we use a heterogenous ensemble of eight classifiers. The classifiers used are the WEKA [7] implementations of k Nearest Neighbour (where k is set through cross validation), Naive Bayes, C4.5 decision tree [13], Support Vector Machines [3] with linear and quadratic basis function kernels, Random Forest [2] (with 100 trees), Rotation Forest [16] (with 10 trees) and a Bayesian network. Each classifier is assigned a weight based on the cross validation training accuracy, and new data are classified with a weighted vote. The set of classifiers were chosen to balance simple and complex classifiers that use probabilistic, tree based and kernel based models. With the exception of k -NN, we do not optimise parameter settings for these classifiers via cross validation. More details are given in [8].

3 Shapelet Transform Refinements

3.1 Binary shapelets

The standard shapelet assessment method measures how well the shapelet splits up all the classes. There are three potential problems with this approach when

classifying multi-class problems. The problems apply to all possible quality measures, but we use information gain to demonstrate the point. Firstly, useful information about a single class may be lost. For example, suppose we have a four class problem and a shapelet produces the order line presented in Figure 1, where each colour represents a different class.

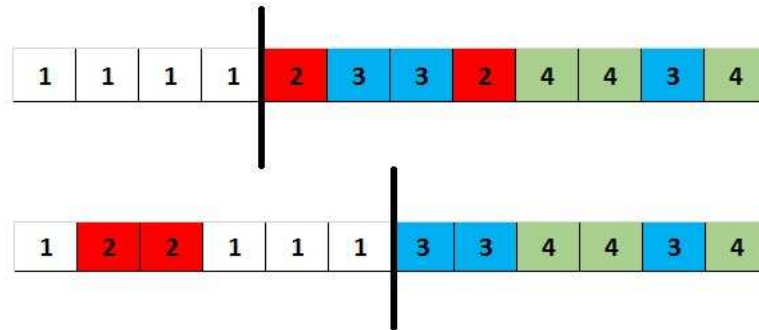


Fig. 1. An example order line split for two shapelets. The top shapelet discriminates between class 1 and the rest perfectly, yet has lower information gain than the orderline shown below it.

The first shapelet groups all of class 1 very well, but cannot distinguish between classes 2, 3 and 4 and hence has a lower information gain than the split produced by the second shapelet in Figure 1 which separates class 1 and 2 from class 3 and 4. The more classes there are, the more likely it is that the quantification of the ability of a shapelet to separate out a single class will be overwhelmed by the mix of other class values. We can mitigate against this potential problem by defining a binary shapelet as one that is assessed by how well it splits the class of the series it originated from from all the other classes. The second problem with searching all shapelets with multi-class assessment arises if one class is much easier to classify than the others. In this case it is likely that more shapelets will be found for easy class than other classes. Although our principle is to find a large number of shapelets (ten times the number of training cases) and let the classifier deal with redundant features, there is still a risk that a large number of similar shapelets for one class will crowd out useful shapelets for another class. If we use binary shapelets we can simply allocate a maximum number of shapelets to each class. We adopt the simple approach of allocating a maximum of k/c shapelets to each class, where c is the number of classes. Finally, the shapelet early abandon described in [17] is not useful for multi-class problems. Given a partial orderline and a split point, the early abandon works by

Algorithm 3 BinaryShapeletSelection(\mathbf{T} , min , max , k)

Input: A list of time series \mathbf{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:  $numClasses \leftarrow getClassDistribution(T)$ 
2:  $kShapeletsMap \leftarrow \emptyset$ 
3:  $prop \leftarrow k/numClasses$ 
4: for all  $T_i$  in  $\mathbf{T}$  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  $kShapeletsMap.asList()$ 
```

upper bounding the information gain by assigning the unassigned series to the side of the split that would give the maximum gain. However, the only way to do this with multi-class problems is to try all permutations. The time this takes quickly rises to offset the possible benefits from the early abandon. If we restrict our attention to just binary shapelets then we can take maximum advantage of the early abandon. The binary shapelet selection is described by Algorithm 3.

3.2 Changing the shapelet evaluation order

Shapelets are phase independent. However, for many problems the localised features are at most only weakly independent in phase, i.e. the best matches will appear close to the location of the candidate. Finding a good match early in $sDist$ increases the likelihood of an early abandon for each $dist$ calculation. Hence, we redefine the order of iteration of the $dist$ calculations within $sDist$ so that we start with the index the shapelet was found at and move consecutively left and right from that point. Figure 2 demonstrates the potential benefit of this approach. The scan from the beginning is unable to early abandon on any of the subseries before the best match. The scan originating at the candidate's location finds the best match faster and hence can early abandon on all the distance calculations at the beginning of the series. Hence, if the location of the best shapelet is weakly phase dependent, we would expect to observe an improvement in the time complexity. The revised function $sDist$, which is a subroutine of $findDistances$ (line 9 in Algorithm 3), is described in Algorithm 4.

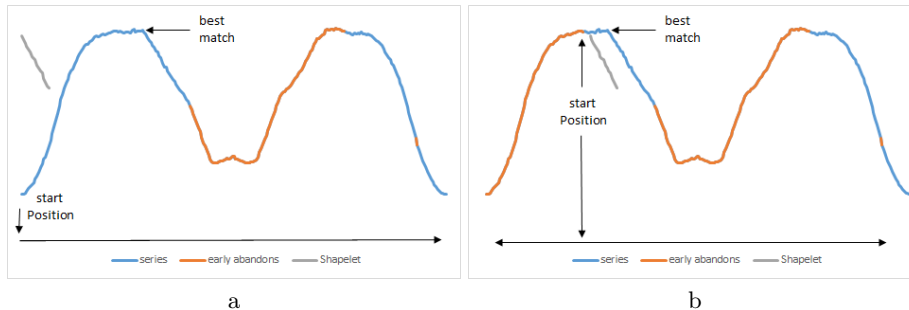


Fig. 2. An example of Euclidean distance early abandon where the $sDist$ scan starts from the beginning (a) and from the place of origin of the candidate shapelet (b). For the scan from the beginning, there are no early abandons until the scan has passed the best match. Because the best match is close to the location of the candidate shapelet, starting from the shapelets original location allows for a greater number of early abandons.

4 Results

We demonstrate the utility of our approach through experiments using 31 benchmark multi-class datasets from UCR [10] and UEA repositories [1]. In common with the vast majority of research in this field, we present results on the standard train/test split. The min and max size of the shapelet transform are found through a train set cross validation described in [9]. As a sanity check, we have also evaluated the binary shapelets on two class problems to demonstrate there is negligible difference. On 16 two class problems, the full transform was better on 7, the binary transform better on 7 and they were tied on 2. All the results and the code to generate them are available from [1].

4.1 Accuracy improvement on multi-class problems

Table 1 gives the results for the full shapelet transform, the binary shapelet transform and standard benchmark classifiers on problems with 3-50 classes. Overall, the binary shapelet transform is better on 19 data sets, the full transform better on 10 and on two they are equal. The difference between the full and the binary shapelet transform is significant at the 5% level using a paired T test and at the 10% level using the binomial and signed rank test. Although neither the binary nor full shapelet transform is significantly more accurate than 1-nearest neighbour with dynamic time warp distance measure (1-NN DTW) and 1-NN DTW with window size set through cross validation (1-NN DTWCV), in a larger study, we have shown that the shapelet transform is significantly better than both 1-NN DTW and 1-NN DTWCV [8]. The main point of including the nearest neighbour results is to highlight the wide variation in accuracy. For example, the shapelet approach is clearly superior on the FacesUCR, fish and CBF problems, but much worse on MiddlePhalanxTW, CricketX and FaceAll.

Algorithm 4 $sDist$ (shapelet S , series T_i)

```
1:  $subSeq \leftarrow getSubSeq(T_i, S.startPos, S.length)$ 
2:  $bestDist \leftarrow euclideanDistance(subSeq, S)$ 
3:  $i \leftarrow 1$ 
4: while  $leftExists \parallel rightExists$  do
5:    $leftExists \leftarrow S.startPos - i \geq 0$ 
6:    $rightExists \leftarrow S.startPos + i < T_i.length$ 
7:   if  $rightExists$  then
8:      $subSeq \leftarrow getSubSeq(T_i, S.startPos + i, S.length)$ 
9:      $currentDist \leftarrow earlyAbandonDistance(subSeq, S, bestDist)$ 
10:    if  $currentDist > bestDist$  then
11:       $bestDist \leftarrow currentDist$ 
12:    if  $leftExists$  then
13:       $subSeq \leftarrow getSubSeq(T_i, S.startPos - i, S.length)$ 
14:       $currentDist \leftarrow earlyAbandonDistance(subSeq, S, bestDist)$ 
15:      if  $currentDist > bestDist$  then
16:         $bestDist \leftarrow currentDist$ 
17:     $i \leftarrow i + 1$ 
18: return  $bestDist$ 
```

This demonstrates the importance of finding the correct transformation space for a given problem.

Figure 3 shows the plot of the difference in accuracy of the full and binary shapelet transform plotted against the number of classes. There is a clear trend of increasing accuracy for the binary transform as the number of classes continues. This is confirmed in Table 2, which presents the same data grouped into bins of ranges of number of classes.

4.2 Accuracy comparison to other shapelet methods

Our goal is to improve the enumerative shapelet transform. Nevertheless, it is informative to compare performance against alternative shapelet based techniques. Table 3 presents the results on 22 multiclass problems for logical shapelets [12], fast shapelets [15] and learning shapelets [5]. We present two sets of results for learning shapelets: those presented on the website (LST1) and those we have recreated using the code available on the website (LST2). Differences between the two may be down to random variation. Overall, there is no significant difference between the classifiers (as tested with Friedman’s rank sum test). A more extensive study is required to determine whether any one of the approaches outperforms the others in terms of accuracy.

4.3 Average case time complexity improvements

One of the benefits of using the binary transform is that it is easier to use the shapelet early abandon described in [12]. Early abandon is less useful when finding the best k shapelets than it is for finding the single best, but when it can

Table 1. Test errors for 5 classifiers on 31 multi class data sets. The binary shapelet and full shapelet both use the weighted ensemble described in Section 2.1. The three nearest neighbour classifiers use Euclidean distance (1NN-ED), dynamic time warping (1NN-DTW) and dynamic time warping with window size set through cross validation (1NN-DTWCV).

dataSet	#classes	Binary Shapelet	Full Shapelet	1NN-ED	1NN-DTW	1-NNDTWCV
fiftywords	50	0.262	0.281	0.63	0.623	0.584
Adiac	37	0.322	0.435	0.369	0.31	0.235
WordSynonyms	25	0.346	0.403	0.253	0.096	0.132
SwedishLeaf	15	0.062	0.093	0.526	0.539	0.539
FaceAll	14	0.246	0.263	0.101	0.05	0.062
FacesUCR	14	0.085	0.087	0.467	0.5	0.5
CricketX	12	0.221	0.218	0.086	0.066	0.065
CricketY	12	0.246	0.236	0.217	0.166	0.166
MedicalImages	10	0.337	0.396	0.35	0.352	0.351
MALLAT	8	0.087	0.06	0.148	0.003	0.006
fish	7	0.006	0.023	0.267	0.267	0.233
Lightning7	7	0.301	0.26	0.183	0.429	0.217
Plane	7	0	0	0.25	0.179	0.214
DistalPhalanxTW	6	0.367	0.367	0.474	0.506	0.506
MiddlePhalanxTW	6	0.409	0.461	0.127	0.137	0.127
ProximalPhalanxTW	6	0.195	0.229	0.087	0.093	0.027
Symbols	6	0.073	0.114	0.121	0.165	0.134
SyntheticControl	6	0.003	0.017	0.127	0.137	0.127
Beef	5	0.1	0.167	0.382	0.351	0.251
Haptics	5	0.497	0.523	0.216	0.17	0.114
Car	4	0.1	0.267	0.231	0.095	0.092
DiatomSizeReduction	4	0.118	0.124	0.425	0.274	0.288
FaceFour	4	0.091	0.057	0.12	0.007	0.017
OliveOil	4	0.067	0.1	0.4	0.35	0.35
Trace	4	0	0.02	0.325	0.105	0.101
ArrowHead	3	0.257	0.229	0.389	0.396	0.391
CBF	3	0.018	0.003	0.426	0.223	0.236
ChlorineConcentration	3	0.311	0.3	0.356	0.208	0.197
DistalPhAgeGroup	3	0.252	0.259	0.038	0	0
MiddlePhAgeGroup	3	0.396	0.37	0.252	0.245	0.201
ProximalPhAgeGroup	3	0.156	0.146	0.252	0.245	0.201
total wins		13	5	2	4	10
Average Rank		2.613	3.06	3.66	3.08	2.58

Table 2. Number of data sets the binary shapelet beats the full shapelet split by number of classes.

Number of classes	Binary Better	Full Better
10 and above	7	2
6 to 9	5	2
3 to 5	7	6
All	19	10

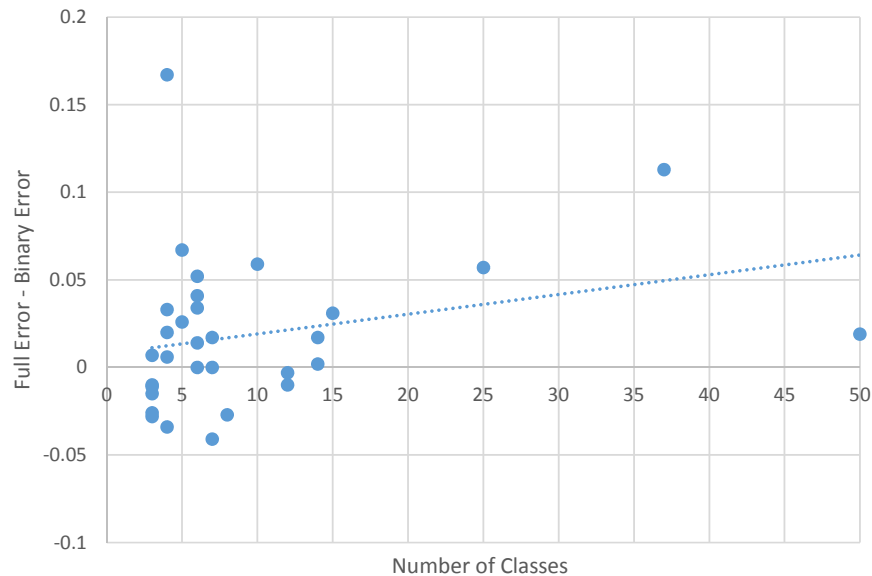


Fig. 3. Number of classes plotted against the difference in error between the full shapelets and the binary shapelets. A positive number indicates the binary shapelets are better. The dotted line is the least squares regression line.

Table 3. Test errors for 6 shapelet based classifiers on 22 multiclass data sets.

Problem	Binary	Full	Logical	Fast	LST1	LST2
Adiac	0.322	0.435	0.414	0.514	0.437	0.737
Beef	0.1	0.167	0.433	0.447	0.24	0.433
CBF	0.018	0.003	0.114	0.053	0.006	0.006
ChlorineConcentration	0.311	0.3	0.382	0.417	0.349	0.406
CricketX	0.221	0.218		0.528	0.209	0.328
CricketY	0.246	0.236		0.52	0.249	0.346
DiatomSizeReduction	0.118	0.127	0.199	0.117	0.033	0
FaceAll	0.246	0.262	0.341	0.411	0.218	0.24
FaceFour	0.091	0.057	0.511	0.09	0.048	0.034
FacesUCR	0.085	0.086	0.338	0.328	0.059	0.092
fiftywords	0.262	0.281		0.489	0.232	0.25
fish	0.006	0.023	0.223	0.197	0.066	0.091
Haptics	0.497	0.519		0.624	0.532	0.5286
Lightning7	0.301	0.26	0.452	0.403	0.197	0.233
MALLAT	0.087	0.06	0.344	0.033	0.046	0.038
MedicalImages	0.337	0.396	0.413	0.433	0.271	0.33
OliveOil	0.067	0.1	0.167	0.213	0.56	0.6
SwedishLeaf	0.062	0.093	0.187	0.269	0.087	0.102
Symbols	0.073	0.114	0.357	0.068	0.036	0.086
SyntheticControl	0.003	0.017	0.53	0.081	0.007	0.023
Trace	0	0.02	0	0.002	0	0
WordSynonyms	0.346	0.401		0.563	0.34	0.409
total wins	8	3	1	1	9	3
Average Rank	2.48	3	4.97	4.73	2.205	3.38

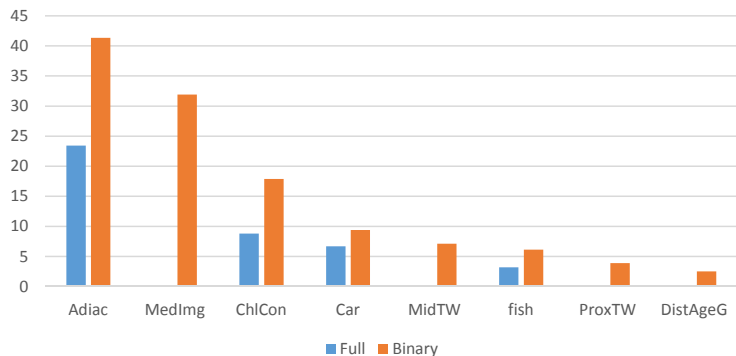


Fig. 4. Number of $sDist$ measurements that were not required because of early abandon (in millions) for both full and binary shapelet discovery on seven datasets.

be employed it can give real benefit. Figure 4 shows that on certain datasets, using the binary shapelet discovery means millions of fewer $sDist$ evaluations.

We assess the improvement from using Algorithm 4 by counting the number of point wise distance calculations required from using the standard approach and the alternative enumeration. For the datasets used in our accuracy experiments, changing the order of enumeration reduces the number of calculations in the distance function by 76% on average. The improvement ranges from negligible (e.g. Lightning7 requires 99.3% of the calculations) to substantial (e.g. Adiac operations count is 63% of the standard approach). This highlights that the best shapelets may or may not be phase independent, but nothing is lost from changing the evaluation order and often substantial improvements are achieved. Full results, all the code and the datasets used can be downloaded from [1].

5 Conclusion

Shapelets are useful for classifying time series where the between class variability can be detected by relatively short, phase independent subseries. They offer an alternative representation that is particularly appealing for problems with long series with recurring patterns. The downside to using shapelets is the time complexity. The heuristic techniques described in recent research [15, 6] offer potential speed up (often at the cost of extra memory) but are essentially different algorithms that are only really analogous to shapelets described in the original research [17]. Our interest is in optimizing the original shapelet finding algorithm within the context of the shapelet transform. We describe incremental improvements to the shapelet transform specifically for multi-class problems. Searching for shapelets assessed on how well they find a single class is more intuitive, faster and becomes more accurate than the alternative as the number of classes increases.

References

1. A. Bagnall. UEA time series classification website. <http://www.uea.ac.uk/computing/tsc>.
2. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
3. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
4. D. Gordon, D. Hendler, and L. Rokach. Fast randomized model generation for shapelet-based time series classification. *arXiv preprint arXiv:1209.5038*, 2012.
5. J. Grabocka. Learning time series shapelets. <http://fs.ismll.de/publicspace/LearningShapelets/>.
6. J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Invariant time-series classification. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.
7. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
8. J. Hills. *Mining Time-series Data using Discriminative Subsequences*. PhD thesis, School of Computing Sciences, University of East Anglia, 2015.
9. J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28:851–881, 2014.
10. E. Keogh and T. Folias. The UCR time series data mining archive. <http://www.cs.ucr.edu/~eamonn/TSDMA/>.
11. J. Lines and A. Bagnall. Alternative quality measures for time series shapelets. In *Intelligent Data Engineering and Automated Learning*, volume 7435 of *Lecture Notes in Computer Science*, pages 475–483. 2012.
12. A. Mueen, E. Keogh, and N. Young. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011.
13. J. R. Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.
14. T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.
15. T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. *Proceedings of the 13th SIAM International Conference on Data Mining*, 2013.
16. J.J. Rodriguez, L.I. Kuncheva, and C.J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.
17. 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):149–182, 2011.