HAL
open science

A Hybrid Approach to Time Series Classification with Shapelets

David Guijo-Rubio, Pedro A Gutiérrez, Romain Tavenard, Anthony Bagnall

To cite this version:

David Guijo-Rubio, Pedro A Gutiérrez, Romain Tavenard, Anthony Bagnall. A Hybrid Approach to Time Series Classification with Shapelets. Intelligent Data Engineering and Automated Learning – IDEAL, Nov 2019, Manchester, United Kingdom. pp.137-144, 10.1007/978-3-030-33607-3_16 . hal-02371422

HAL Id: hal-02371422

https://hal.science/hal-02371422

Submitted on 19 Nov 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A hybrid approach to time series classification with shapelets

David Guijo-Rubio[1,2], Pedro A. Gutiérrez[1], Romain Tavenard[3] and Anthony Bagnall[2]

[1] Department of Computer Sciences, Universidad de Córdoba, Córdoba, Spain
[2] University of East Anglia, Norwich, UK, NR47TJ.
[3] Univ. Rennes, CNRS, LETG/IRISA

Abstract. Shapelets are phase independent subseries that can be used to discriminate between time series. Shapelets have proved to be very effective primitives for time series classification. The two most prominent shapelet based classification algorithms are the shapelet transform (ST) and learned shapelets (LS). One significant difference between these approaches is that ST is data driven, whereas LS searches the entire shapelet space through stochastic gradient descent. The weakness of the former is that full enumeration of possible shapelets is very time consuming. The problem with the latter is that it is very dependent on the initialisation of the shapelets. We propose hybridising the two approaches through a pipeline that includes a time constrained data driven shapelet search which is then passed to a neural network architecture of learned shapelets for tuning. The tuned shapelets are extracted and formed into a transform, which is then classified with a rotation forest. We show that this hybrid approach is significantly better than either approach in isolation, and that the resulting classifier is not significantly worse than a full shapelet search.

Keywords: Time series classification · Shapelets · Convolutional Neural Networks.

1 Introduction

Shapelets [1] are discriminatory phase independent subsequences that form a basic primitive in many time series algorithms. For classification, shapelets are assessed using their distance to train set time series and the usefulness of these distances in discriminating between classes. Shapelet based features define a distinct form of discrimination which can be characterised as quantifying whether a particular shape exists in a series or not (at any location). Shapelets have proved an effective tool for classification [2] and have been a popular research topic. One key distinction between research threads is whether shapelets are extracted from the training data or whether the space of all possible shapelets is searched. The data driven approach was used in early work with shapelets [1,3,4] and has been employed to find effective classifiers [2,5]. The learned shapelet

approach [6] was the first search based algorithm. Later work has bridged the gap between the learning shapelets and convolutional neural networks (CNN) through defining each shapelet as a variant of a convolutional filter [7]. Our aim is to investigate whether we can create a better classifier by hybridising data driven search and stochastic gradient descent learning. Our approach is to randomly sample shapelets in the data for a fixed time using the method described in [5], retain the best k found, then tune these shapelets with the learning shapelet algorithm [6] implemented using a neural network library. We test this on data from the UCR archive [8]. We demonstrate that, under controlled parameterisation, this approach is better than either algorithm in isolation. Furthermore, we show that a one hour search followed by tuning is not significantly worse than the reported results using the full shapelet transform [9].

The rest of the paper is structured as follows. In Section 2 we provide an overview of the shapelet transform and learning shapelets algorithm. In Section 3 we describe how we have hybridised the two approaches. Results are described in Section 4, and we conclude in Section 5.

2 Shapelet Background

We denote a vector in bold and a matrix in capital bold. A case/instance is a pair $\{\mathbf{x}, y\}$ with m observations $x_1, \ldots, x_m$ (the time series) and discrete class variable y with c possible values. A list of n cases with associated class labels is $\mathbf{T} = \langle \mathbf{X}, \mathbf{y} \rangle = \langle (\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n) \rangle$. A shapelet $\mathbf{s}$ is a time series $\langle s_1, \ldots, s_l \rangle$ where $l \leq m$. Shapelet based classification requires a method of finding and assessing shapelets, then an algorithm for using the selected shapelets for classification. All shapelet finding algorithms require the measuring of the distance between a candidate and a time series. This is done by sliding the candidate along the series and calculating the Euclidean distance at each position (after normalisation) to find the minimum. The distance between a shapelet $\mathbf{s}$ and a time series case is then given by Equation 1,

$$sDist(\mathbf{s}, \mathbf{t}) = \min_{\mathbf{w} \in \mathbf{W}} (dist(\mathbf{s}, \mathbf{w})), \tag{1}$$

where $\mathbf{W}$ is the set of all subsequences which are the same length as $\mathbf{s}$ in $\mathbf{t}$, and *dist* is the Euclidean distance between two equal length series. The original shapelet algorithm [1] constructed a decision tree by finding a shapelet at each node. Subsequent research [10] demonstrated it was better to use shapelets as a transformation. We can summarise the shapelet transform (ST) approach as: search for the best k shapelets in the data; transform the data so that each new attribute j represents distance between series i and shapelet j; construct a standard classifier on the resulting transformed data. It has been shown that there is little need to enumerate the possible space of shapelets in the data. Random sampling of a tiny proportion of the shapelet space does not lead to a significant decrease on accuracy [5]. The current shapelet transform algorithm can be configured so that it searches for shapelets for a maximum amount of

time. We call this a contract classifier, since it is given a time contract it must fulfill.

The ST pipeline can be summarised as follows:

1. **Search**: randomly sample shapelets from the train data for a fixed amount of time, keeping the best.
2. **Transform**: create new train and test data where attributes are distances between instances and shapelets (see Equation 1).
3. **Fit Model**: build the classifier on the train data.
4. **Predict**: estimate class values on the test data.

Another alternative to the enumeration of all possible shapelets is to use optimization techniques to learn good shapelets. Learning shapelets (LS [6]) algorithm involves learning shapelet values as model parameters rather than directly extracting them from the data. Hence, resulting shapelets are no longer subseries from the training set. The LS algorithm adopts an initialisation stage to find k shapelets through clustering shapelets observed in the data. It then jointly learns the weights for a regularised logistic regression and the shapelet set using a two stage iterative process for Shapelet Tranform representation to feed a final logistic regression classifier. The LS pipeline can be summarised as follows:

1. **Initialise**: find initial shapelets from the train data through clustering subsequences and initialise model weights.
2. **Fit Model**: For a given maximum number of iterations:
   (a) **Update Loss**: adjust loss function.
   (b) **Update Model Weights**: adjust weights to minimize loss.
   (c) **Update Shapelets**: adjust shapelets to mimize loss.
3. **Predict**: estimate class values on the test data.

An experimental comparison of these algorithms found ST to be significantly more accurate than LS [2]. LS suffers from three related problems that may have caused this difference. Firstly, LS is very sensitive to the initialisation of shapelets, and, secondly, the clustering algorithm adopted to partially overcome this problem is memory intensive. Finally, the implementation is very complex; despite communications with the LS author, we cannot rule out there being bugs in the Java implementation[4]. One way of possibly mitigating these problems is to use stable open source software.

3 Hybrid Shapelet Classifier

A shapelet distance $sDist$ is evaluated by sliding the shapelet to each series as would be done for a convolution filter (see Equation 1). The LS model can be

[4] https://github.com/TonyBagnall/uea-tsc

be implemented as a variant of a CNN [7] within a standard neural network framework. A neural network model is defined that takes time series as inputs and outputs class probabilities.

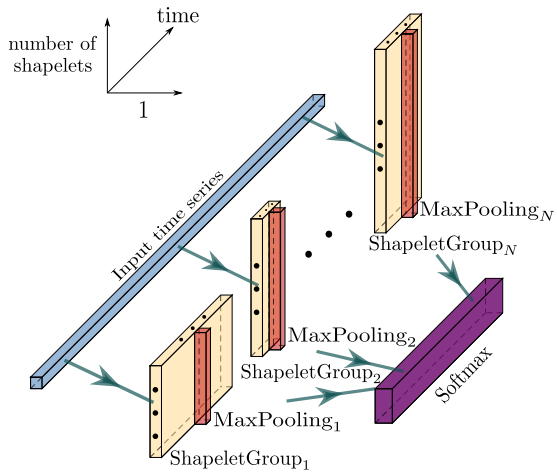


Fig. 1. Learning Time Series Shapelet (LS) model as a neural network architecture.

As shown in Figure 1, this model is composed of a first layer (called the shapelet layer hereafter) that extracts a ST-like representation which then feeds into a logistic regression layer. In practice, the shapelet layer is made of several shapelet blocks (one block per shapelet length). Each shapelet block can be decomposed into:

1. a feature extraction step that computes pairwise distances $dist(\mathbf{s}, \mathbf{w})$ between the considered shapelets and all the subsequences with the same length as $\mathbf{s}$ in $\mathbf{t}$ and
2. a pooling step that retains the minimum of all distances.

Note that this is very similar in spirit to a convolutional layer that would compute dot products between filters and all the subsequences in $\mathbf{t}$ which would be typically followed by a (max-)pooling layer. Finally, the optimization procedure consists in tuning both the shapelet values and the parameters of the logistic regression through stochastic gradient descent.

There are two ways we could combine the approaches: we could use the transform to search for a better starting point for the LS classification algorithm or use the LS algorithm to tune the shapelets found by ST, but retain the classification method in ST. The first approach involves replacing the **initialise** stage in LS with a time constrained **search**, then proceeding with LS as normal. The second approach is to perform a tuning stage with LS' **fit model** between **search** and **transform** from ST. We evaluate both approaches and consider three classifiers:

1. **ST-RF**: Shapelet transform contracted for one hour or ten hours, then build and evaluate a rotation forest classifier on the transformed data
2. **Hybrid-LR**: Use the shapelets found for ST as an initialisation for the neural network (LS model), then use the final logistic regression classifier on the test data.
3. **Hybrid-RF**: As Hybrid-LR, but rather than use the logistic regression, use rotation forest as a classifier, as with ST.

We use the rotation forest classifier [11] with fixed parameters for ST and Hybrid-RF, since it has been shown to be very good at problems with continuous attributes [12].

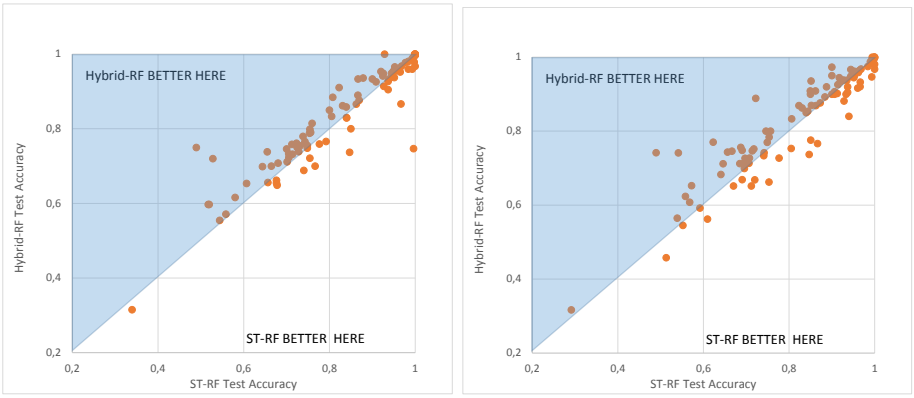
4 Results

The rotation forest has 200 trees, uses a group size of 3 and class selection probability of 0.5. More details can be found in [12]. We set ST to find a maximum of 100 shapelets in either one hour or ten hours. The LS model is optimised using the Adagrad adaptive gradient optimisation algorithm [13]. The learning rate parameter is fixed to 0.1 and the training will be run for 2000 epochs for all datasets. Moreover the ℓ_2 regularisation parameter over classification weights is 0.01 and the batch size is fixed to 128. Note that for this first approximation, all the parameters are fixed to default values for simplicity and to reduce the computational cost.

We evaluate the shapelet approaches on a subset of 92 data of the 128 UCR data [8]. The data that are omitted are done so for practical and implementation reasons. 14 of the 128 data have missing values or are unequal length and our system is not able to handle this characteristic. The LS is unable to handle 22 of the remaining 114 data because of memory constraints; long shapelets require a large amount of memory. All reported results are on the standard train test splits. All learning and tuning are conducted exclusively on the train data, and accuracy is assessed on the unseen test split.

We are interested in testing whether tuning the shapelets after a data driven search significantly improves the overall classifier performance. To do this, we control other factors that cause variation. We have intentionally set up ST-RF and Hybrid-RF so that the only difference between the two experiments is the tuning stage. Any improvement can be attributed to this, since in all other ways ST-RF and Hybrid-RF are identical.

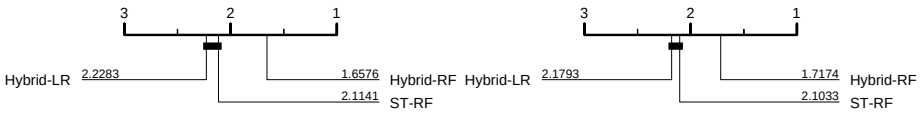
Figure 2(a) shows the scatter plot of test accuracy for ST-RF and Hybrid-RF for a one hour shapelet search. Tuning makes a significant difference: 51 datasets have improved accuracy, whereas just 32 have decreased performance (with 9 ties). The test of difference is significant with a binomial test, a paired T test and a Wilcoxon sign rank test. Figure 2(b) shows the same results for a 10 hour shapelet search. The pattern of results is the same; tuning improves accuracy on 52 problems, makes things worse on 32 and makes no difference on 8. The difference is also significant.



(a) 1 hour shapelet transform search (b) 10 hours shapelet transform search

Fig. 2. Test accuracy for the shapelet transform before and after tuning with LS.

Whilst significant, it is worth noting that the improvement is not guaranteed. As a sanity check, it is worth checking whether it is better to use the logistic regression from LS as a classifier itself rather than extracting the shapelets for use with rotation forest. Figure 3 shows the critical difference diagrams [14] for the ST and Hybrid-RF transform in comparison to those found with Hybrid-LR. It demonstrates two things: firstly, tuning the shapelets then classifying with the LS model (Hybrid-LR) makes no significant difference when compared to ST alone; and secondly, that extracting shapelets from LS and using a rotation forest (Hybrid-RF) is significantly better than both a logistic regression classifier (Hybrid-LR) and using rotation forest with a transform generated by untuned shapelets (ST-RF).



(a) 1 hour shapelet search (b) 10 hours shapelet search

Fig. 3. Critical difference diagrams for three classifiers. The classifiers are: shapelet transform with rotation forest (ST); Learning Time Series Shapelets using the transform shapelets as a starting point (Hybrid-LR); and shapelet transform on shapelets extracted from LS with a rotation forest (Hybrid-RF).

For context, it is useful to compare the results to previously published results. ST is a key component of the meta-ensemble HIVE-COTE [9]. The ST results for HIVE-COTE were found through a computationally expensive enumeration of all possible shapelets. Figure 4 shows that we can achieve results that are not significantly worse than the enumerative search by simply tuning the one hour search shapelets.

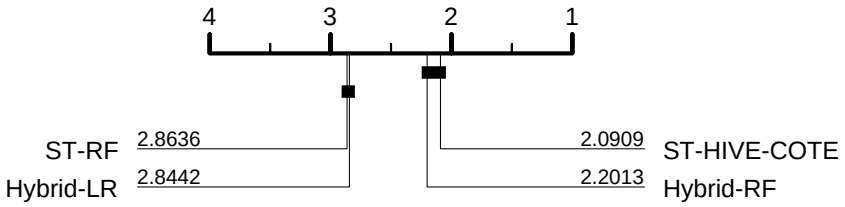


Fig. 4. Comparison of performance of three classifiers based on a one hour shapelet search with an exhaustive search based shapelet classifier presented in [9].

5 Conclusion

We have demonstrated that the core concept of tuning shapelets found in the data with a gradient descent algorithm has merit. Tuning significantly improved accuracy after both a one hour search and a ten hour search. Indeed, the improvement was enough to achieve results not significantly worse than the state-of-the-art shapelet approach. This is surprising and exceeded our expectation. However, these experiments have limitations. Firstly, the neural network implementation of learning shapelets is very memory intensive. This means we have not been able to compare against using more shapelets in the transform nor with all the datasets. Nor have we been able to evaluate on resamples rather than a single train test split. Secondly, whilst comparing on all problems is desirable to remove any suspicion of cherry picking, the presence of many smaller problems may mask the benefit of the full enumeration: many of the problems can be fully enumerated in one hour. Thirdly, we have not described the training time for the LS model in our results, because at the time of writing we do not have an integrated solution: we search for shapelets in the Java version [15] on CPU and train LS using a dedicated pytorch implementation on GPU. This makes timing comparisons problematic. A fully integrated version is in development using the sktime toolkit [16]. Results and code are available from the associated website [5]. Despite these limitations, these results are promising and support the central hypothesis than tuning can improve shapelets found in the data. The next stage is to evaluate the methods on larger problems and to assess the relative merits of greater search time, retaining more shapelets and selective tuning. Furthermore, an iterative search and tune algorithm may prove better than our current sequential model of search then tune.

Acknowledgement. This research has been partially supported by the Ministerio de Economía, Industria y Competitividad of Spain (Grant Refs. TIN2017-85887-C2-1-P and TIN2017-90567-REDT) as well as Agence Nationale de la Recherche through MATS project (ANR-18-CE23-0006). D. Guijo-Rubio's research has been supported by the FPU Predoctoral and Short Placements Programs from Ministerio de Educación y Ciencia of Spain (Grants Ref. FPU16/02128

[5] http://www.timeseriesclassification.com/hybrid.php

and EST18/00280, respectively). Some experiments used a Titan X Pascal donated by the NVIDIA Corporation.

References

1. L. Ye and E. Keogh, "Time series shapelets: a novel technique that allows accurate, interpretable and fast classification," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, pp. 149–182, 2011.
2. 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 advances," *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, 2017.
3. 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.
4. 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.
5. A. Bostrom and A. Bagnall, "Binary shapelet transform for multiclass time series classification," *Transactions on Large-Scale Data and Knowledge Centered Systems*, vol. 32, pp. 24–46, 2017.
6. 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.
7. R. Tavenard, "tslearn: A machine learning toolkit dedicated to time-series data." `https://github.com/rtavenar/tslearn`, 2017.
8. H. Dau, A. Bagnall, K. Kamgar, M. Yeh, Y. Zhu, S. Gharghabi, and C. Ratanamahatana, "The UCR time series archive," *ArXiv e-prints*, vol. arXiv:1810.07758, 2018.
9. J. Lines, S. Taylor, and A. Bagnall, "Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles," *ACM Trans. Knowledge Discovery from Data*, vol. 12, no. 5, 2018.
10. J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining and Knowledge Discovery*, vol. 28, no. 4, pp. 851–881, 2014.
11. J. Rodriguez, L. Kuncheva, and C. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006.
12. A. Bagnall, A. Bostrom, G. Cawley, M. Flynn, J. Large, and J. Lines, "Is rotation forest the best classifier for problems with continuous features?," *ArXiv e-prints*, vol. arXiv:1809.06705, 2018.
13. J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.
14. J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
15. uea tsc, "A weka compatible toolkit for time series classification and clustering." `https://github.com/TonyBagnall/uea-tsc`, 2019.
16. sktime, "A toolbox for data science with time series." `https://github.com/alan-turing-institute/sktime`, 2019.