Noname manuscript No.
(will be inserted by the editor)

Fast Classification of Univariate and Multivariate Time series Through Shapelets Discovery

Josif Grabocka · Martin Wistuba · Lars Schmidt-Thieme

Received: Dec 17, 2014 / Revised: Aug 26, 2015 / Accepted: Nov 07, 2015

Abstract Time-series classification is an important problem for the data mining community due to the wide range of application domains involving time-series data. A recent paradigm, called shapelets, represents patterns that are highly predictive for the target variable. Shapelets are discovered by measuring the prediction accuracy of a set of potential (shapelet) candidates. The candidates typically consist of all the segments of a dataset, therefore, the discovery of shapelets is computationally expensive. This paper proposes a novel method that avoids measuring the prediction accuracy of similar candidates in Euclidean distance space, through an online clustering/pruning technique. In addition, our algorithm incorporates a supervised shapelet selection that filters out only those candidates that improve classification accuracy. Empirical evidence on 45 univariate datasets from the UCR collection demonstrate that our method is 3-4 orders of magnitudes faster than the fastest existing shapelet-discovery method, while providing better prediction accuracy. In addition, we extended our method to multivariate time-series data. Runtime results over 4 real-life multivariate datasets indicate that our method can classify MB-scale data in a matter of seconds and GB-scale data in a matter of minutes. The achievements do not compromise quality, on the contrary, our method is even superior to the multivariate baseline in terms of classification accuracy.

Keywords Time-series classification; Multivariate time series; Shapelet discovery

Josif Grabocka, Martin Wistuba, Lars Schmidt-Thieme
Information Systems and Machine Learning Lab
University of Hildesheim, 31141 Hildesheim, Germany
E-mail: {josif,wistuba,schidt-thieme}@ismll.uni-hildesheim.de

1 Introduction

Classification of time-series data has attracted considerable interest in the recent decades, which is not surprising given the numerous domains where time series are collected. A recent paradigm has emerged into the perspective of classifying time series, the notion of *shapelets*. Shapelets are supervised segments of series that are highly descriptive of the target variable (Ye and Keogh, 2009). In the recent years, shapelets have achieved a high momentum in terms of research focus (Ye and Keogh, 2009; Zakaria, Mueen and Keogh, 2012; Mueen, Keogh and Young, 2011; Rakthanmanon and Keogh, 2013).

Distances of time series to shapelets can be perceived as new classification predictors, also called *"the shapelet-transformed data"* (Hills, Lines, Baranauskas, Mapp and Bagnall, 2013). It has been shown by various researchers that shapelet-derived predictors boost the classification accuracy (Ye and Keogh, 2011; Mueen et al., 2011). In particular, shapelets are efficient in datasets where the class discrimination is attributed to local variations of the series content, instead of the global structure (Ye and Keogh, 2009). Even though not explicitly mentioned by the related work, the discovery of shapelets can be categorized as a supervised dimensionality reduction technique. In addition, shapelets also provide interpretive features that help domain experts understand the differences between the target classes.

In contrast to the high classification accuracy, discovering shapelets remains challenging in terms of runtime. The current discovery methods need to search for the most predictive shapelets from all the possible segments of a time-series dataset (Ye and Keogh, 2009; Mueen et al., 2011). Since the number of possible candidates is high, the required time for evaluating the prediction quality of each candidate is prohibitive for large datasets. Therefore, the time-series research community has proposed several speed-up techniques (Ye and Keogh, 2009; Mueen et al., 2011; Rakthanmanon and Keogh, 2013), aiming at making shapelet discovery **feasible** in terms of time.

This paper proposes a novel method that discovers time-series shapelets considerably faster than the fastest existing method. Our method follows the knowledge that time-series instances contain lots of similar segments. Often inter-class variations of time series depend on differences within small segments, with the remaining parts of the series being similar. Therefore, we hypothesize that the time needed to discover shapelets can be **scaled**-up by pruning candidate segments that are similar in Euclidean distance space. We introduce a fast distance-based clustering approach to prune future segments that result similar to previously considered ones. In addition, we propose a fast supervised selection of shapelets that filters out the qualitative shapelets using an incremental nearest-neighbor classifier. Extensive experiments conducted on real-life data demonstrate a large reduction (3-4 orders of magnitude) of the discovery time, by even gaining prediction accuracy with respect to baselines. The contributions of this paper can be short-listed as follows:

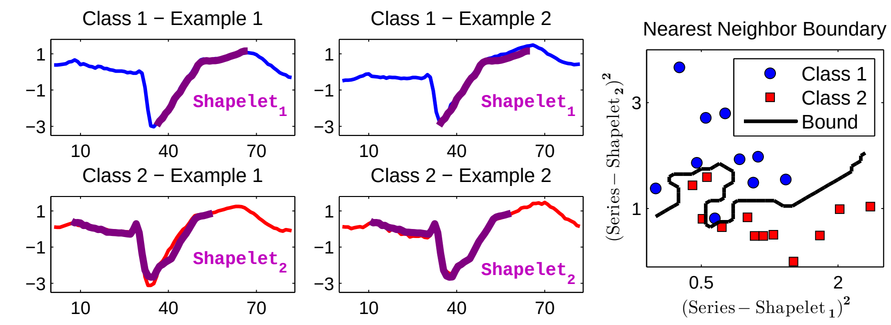


Fig. 1 TwoLeadECG dataset: Aligning shapelets to the closest series segments, and on right the resulting 2-dimensional shapelet-transformed training data.

1. A fast pruning strategy for similar shapelets in Euclidean space involving a distance-based clustering approach;
2. A fast supervised selection of qualitative shapelets using an incremental nearest-neighbor classifier, conducted jointly with the pruning;
3. Extensive experimental results against the fastest existing univariate shapelet discovery methods on a large set of 45 time-series datasets.
4. Extension to multivariate time-series datasets showing that our method scales to GB-sized data

2 Related Work

Shapelets were introduced by (Ye and Keogh, 2009) as a new primitive representation of time series that is highly predictive of the target. A large pool of candidates from all segments of a dataset were assessed as potential shapelet candidates, while the minimum distance of series to shapelets was used as a predictive feature. The best performing candidates were ranked using the information gain criteria over the target. Successively, other prediction quality metrics were also proposed such as the Kruskal-Wallis or Mood's median (Hills et al., 2013), as well as F-Stats (Lines and Bagnall, 2012). The minimum distance of the time series to a set of shapelets can be understood as a data transformation (dimensionality reduction) and is called *shapelet-transformed data* (Hills et al., 2013). Certain classifiers, such as SVMs or rotation forests, have been shown to perform competitively over the shapelet-transformed predictors (Hills et al., 2013).

The excessive amount of potential candidates makes brute-force (exhaustive) shapelet discovery intractable for large datasets. Therefore, researchers have come up with various approaches for speeding up the search. Early abandoning of the Euclidean distance computation combined with an entropy pruning of the information gain metric is an early pioneer in that context (Ye and Keogh, 2009). Additional papers emphasize the reuse of computations and the pruning of the search space (Mueen et al., 2011), while the projection of series to SAX representation was also elaborated (Rakthanmanon and Keogh, 2013).

Furthermore, the discovery time of shapelets has been minimized by mining infrequent shapelet candidates (He, Zhuang, Shang, Shi et al., 2012). Speed-ups have also been attempted by using hardware-based implementations, such as the usage of the processing power of GPUs for reducing search time (Chang, Deka, Hwu and Roth, 2012).

In terms of applicability, shapelets have been utilized in a battery of real-life domains. Unsupervised shapelets discovery, for instance, has been shown useful in clustering time series (Zakaria et al., 2012). Shapelets been used in classifying/identifying humans through their gait patterns (Sivakumar and Shajina, 2012). Gesture recognition is another application domain where the discovery of shapelets has played an instrumental role in improving the prediction accuracy (Hartmann and Link, 2010; Hartmann, Schwab and Link, 2010). In the realm of medical and health informatics, interpretable shapelets have been shown to help in early classification of time series (Xing, Pei, Yu and Wang, 2011; Xing, Pei and Yu, 2012).

In contrast to the state-or-the-art methods, we propose a fast novel method that discovers shapelets by combining a direct similarity-based pruning strategy of candidates with an incremental classification technique.

3 Scalable Shapelet Discovery

Table 1 Summary of Notations

Symbol $\in$ Domain	**Description**
$N \in \mathbb{N}$	Number of time series
$Q \in \mathbb{N}$	Length of time series
$T \in \mathbb{R}^{N \times Q}$	Time-series datasets
$\Phi \in \mathbb{R}^{R}$	Set of shapelet lengths
$s \in \mathbb{R}^{\Phi_*}$	A shapelet candidate
$\mathcal{D} \in \left(\mathbb{R}^{L} \times \mathbb{R}^{L}\right) \to \mathbb{R}$	Distance between a shapelet and a segment
$R \in \mathbb{N}$	Number of different candidate lengths
$p \in [1, \ldots, 100]$	Pruning distance percentile
$r \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$	Dimensionality reduction ratio
$X \in \mathbb{R}^{N \times N}$	Pairwise distances between series
$Y \in \mathbb{N}^{N}$	Labels of the series
$\mathcal{A} \in \mathbb{R}^{* \times *}$	Accepted shapelet candidates
$\mathcal{R} \in \mathbb{R}^{* \times *}$	Rejected shapelet candidates
$V \in \mathbb{N}$	Number of dimensions
$\lvert s \rvert \in \mathbb{N}$	Length (cardinality) of shapelet s

3.1 Distances of Shapelets to Series as Classification Features

Throughout this paper we denote a time-series dataset having N series of Q points each, as $T \in \mathbb{R}^{N \times Q}$. While our method can work with series of arbitrary lengths, we define a single length Q for ease of mathematical formalism.

The distances of shapelets to series can be used as classification features, also known as shapelet-transformed features (Hills et al., 2013). The distance of a candidate shapelet to the closest segment of a series can be perceived as a membership degree for that particular shapelet. Equations 1 and 2 formalize the minimum distances between a shapelet s and the dataset T as a vector of the Euclidean distances ($\mathcal{D}$) between the shapelet and the closest segment of each series. (The notation $V_{a:b}$ denotes a sub-sequence of vector V from the a-th element to the b-th element.)

$$\mathbf{MinDist}(s,T) := \begin{bmatrix} \mathcal{D}(s,T_1) \\ \mathcal{D}(s,T_2) \\ \vdots \\ \mathcal{D}(s,T_N) \end{bmatrix} \tag{1}$$

$$\mathcal{D}(s,T_i) := \min_{j=1,\ldots,Q-|s|+1} \left\| T_{i,j:j+|s|-1} - s \right\|^2 \tag{2}$$

An illustration of the minimum distances between shapelets and series is shown in Figure 1 for the TwoLeadECG dataset. Two shapelets (purple) are matched to four time series of two different classes (red and blue). Following the principle that Equation 2 states, the distance of a shapelet is computed to the closest series segment. The distances between training time series and the two shapelets can project the dataset to a 2-dimensional shapelet-transformed space, as shown on the right sub-plot. A nearest neighbor classifier and the corresponding classification decision boundary is also illustrated.

3.2 Quantification of Similarity Using a Distance Threshold

A time-series dataset generally contains lots of similar patterns spread over various instances. Since series from the same class generally follow a similar structure, similar patterns repeat over time series of the same class. Similarities can also be observed among time series of different classes, because often classes are discriminated by differences in small sub-sequences rather than the global structure. As a result, we raise the hypothesis that existing state-of-the-art techniques, which exhaustively search all candidates, inefficiently consider lots of very similar patterns.

Figure 2 illustrates the distribution of distances among arbitrary pairs of candidate segments from various time series of the UCR collection of datasets (Keogh, Zhu, Hu, Y., Xi, Wei and Ratanamahatana, 2011). As can be seen from sub-figure *a)*, the distribution of distances is highly skewed towards zero, which indicate that most candidates are very similar to each other. However, a threshold separation on the similarity distance is required to judge segments as being similar or not. We propose to use a threshold over the percentile on the distribution of distances. For instance, Figure *2.b)* displays pairs of similar segments whose pairwise distances are within the 25-th percentile of the distance distribution.

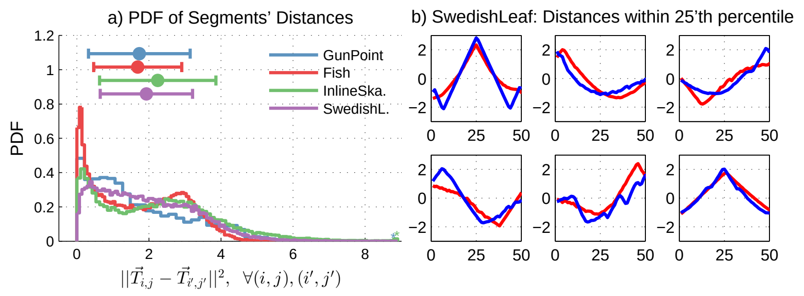


Fig. 2 **a)**Distribution of distances among random pairs of candidates; **b)** Illustration of similar segments from the SwedishLeaf dataset with pairwise distances less than the 25−th percentile of the distribution in **a)**.

Algorithm 1: ComputeThreshold: Compute the pruning similarity distance threshold ϵ.

```
    Data: Time series data $T \in \mathbb{R}^{N \times Q}$, Percentile $p \in [1, \ldots, 100]$, Shapelet Lengths
          $\Phi \in \mathbb{N}^R$
    Result: Threshold distance $\epsilon \in \mathbb{R}$
1   $Z \leftarrow \emptyset$;
2   for $1, \ldots, NQ$ do
3       Draw random shapelet length $\Phi_r \sim \mathcal{U}(\Phi_1, \ldots, \Phi_R)$ ;
4       Draw segment indices $(i, j) \sim (\mathcal{U}(1, \ldots, N), \mathcal{U}(1, \ldots, Q - \Phi_r + 1))$ ;
5       Draw segment indices $(i', j') \sim (\mathcal{U}(1, \ldots, N), \mathcal{U}(1, \ldots, Q - \Phi_r + 1))$ ;
6       $Z \leftarrow Z \cup \left\{ \frac{1}{\Phi_r} ||T_{i,j:j+\Phi_r-1} - T_{i',j':j'+\Phi_r-1}||^2 \right\}$;
7   end
8   $Z \leftarrow \text{sort}(Z)$;
9   $\epsilon \leftarrow Z_{\lceil \frac{p}{100} N Q \rceil}$;
10  return $\epsilon$
```

The procedure of determining a distance threshold value, denoted ϵ and belonging to the p-th percentile of the distance distribution, is described in Algorithm 1. The algorithm selects a pair of random segments starting at indices $(i, j), (i', j')$ and having random shapelet lengths Φ_r. Then a distribution is built by accumulating the distances of random pairs of segments and the distance value that corresponds to the desired percentile p is computed from the sorted list of distance values. For instance, in case all the distance values are sorted from smallest to largest, then the 25-th percentile is the value at the index that belongs to 25% of the total indices.

In total, there are $\mathcal{O}(NQR)$ segments in a time-series dataset and the total number of pairs is $\frac{1}{2}(NQR)(NQR-1)$. However, in order to estimate the distribution of a set of values (here distances), one does not need to have access to the full population of values. On the contrary, a sample of values are sufficient for estimating the distribution. In order to balance between a fast and accurate compromise we choose to select NQ-many random segment pairs for estimating the distance distributions. The runtime speed up success of Section 4.3 indicates that the distance threshold estimation is accurate.

Algorithm 2: DiscoverShapelets: Scalable discovery of shapelets

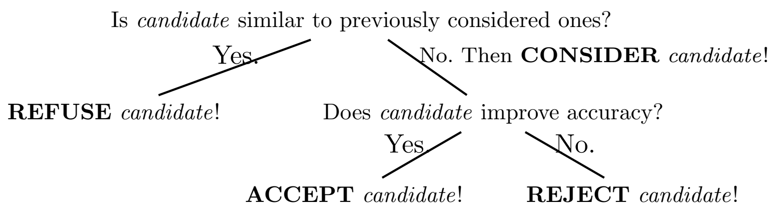
```
Data: Time series data $T \in \mathbb{R}^{N \times Q}$, Labels $Y \in \mathbb{N}^N$ Distance Threshold Percentile
      $p \in [1, \dots, 100]$, Piecewise Aggregate Approximation ratio $r \in \{\frac{1}{2}, \frac{1}{4}, \dots\}$,
      Shapelet lengths $\Phi \in \mathbb{N}^R$
Result: Accepted shapelets list $\mathcal{A} \in \mathbb{R}^{* \times *}$, Minimum Distances $D \in \mathbb{R}^{* \times *}$
1  $\epsilon \leftarrow$ ComputeThreshold$(T, p, \Phi)$;
2  $\mathcal{A} \leftarrow \emptyset, \mathcal{R} \leftarrow \emptyset, D \leftarrow \emptyset, X \leftarrow \mathbf{0}_{N \times N}$, prevAccuracy $\leftarrow -\infty$;
3  for $1, \dots, NML$ do
4      Draw random series: $i \sim \mathcal{U}\{1, \dots, N\}$;
5      Draw random shapelet length: $\Phi_r \sim \mathcal{U}\{\Phi_1, \dots, \Phi_R\}$;
6      Draw random segment start: $j \sim \mathcal{U}\{1, \dots, Q - \Phi_r + 1\}$;
7      Selected random candidate: $s \leftarrow T_{i, j:j+\Phi_r-1}$;
8      if $\neg LookUp(s, \mathcal{A}, \epsilon) \wedge \neg LookUp(s, \mathcal{R}, \epsilon)$ then
9          $d^s \leftarrow$ MinDist$(s, T)$ ;
10         for $i = 1, \dots, N;\ m = i + 1, \dots, N$ do
11             $X_{i,m} \leftarrow X_{i,m} + \left(d_i^s - d_m^s\right)^2$;
12         end
13         $\alpha \leftarrow$ Accuracy$(X, Y)$;
14         if $\alpha > prevAccuracy$ then
15             $\mathcal{A} \leftarrow \mathcal{A} \cup \{s\}$;
16             $D \leftarrow D \cup \{d^s\}$;
17             prevAccuracy $\leftarrow \alpha$;
18         else
19             $\mathcal{R} \leftarrow \mathcal{R} \cup \{s\}$;
20             for $i = 1, \dots, N;\ m = i + 1, \dots, N$ do
21                 $X_{i,m} \leftarrow X_{i,m} - \left(d_i^s - d_m^s\right)^2$;
22             end
23         end
24     end
25 end
26 return $\mathcal{A}, D$
```

3.3 Main Method: Scalable Discovery of Time-series Shapelets

The scalable discovery of time-series shapelets follows the two primary principles of this paper: *i)* Pruning of similar candidates, and *ii)* on-the-fly supervised selection of shapelets. The rationale of these principles is based on the knowledge that the majority of patterns from any specific time series are similar to patterns in other series of the same dataset. Therefore, it is computationally non-optimal to measure the quality of lots of very similar candidates. Instead, we aim at considering only a small nucleus of non-redundant candidates.

3.3.1 Taxonomy of The Terms

The fate of any candidate shapelet will be one of **refused**, **considered**, **accepted** and **rejected**. The decision tree below helps clarifying those terms.


Is *candidate* similar to previously considered ones?
Yes.
No. Then **CONSIDER** *candidate*!
REFUSE *candidate*!
Does *candidate* improve accuracy?
Yes
No.
ACCEPT *candidate*!
REJECT *candidate*!

The similarity of a candidate is first evaluated by looking up whether a close candidate has been previously considered, i.e has been previously flagged as either accepted or rejected. The considered non-redundant (non-similar to previous) candidates are subsequently checked on whether they improve the classification accuracy of previously selected candidates, and are either marked as accepted or rejected.

We are presenting our method as Algorithm 2 and incrementally walking the reader through the steps. The algorithm is started by compressing the time series via the Piecewise Aggregate Approximation technique, to be detailed in Section 3.4. In order to prune similar candidates, the threshold distance ϵ is computed using Algorithm 1. Our method operates by populating two lists of accepted and rejected shapelets, denoted as $\mathcal{A}$ and $\mathcal{R}$, and storing a distance matrix X for distances between series in the shapelet-transformed space.

3.3.2 Pruning Similar Candidates

Random shapelet candidates, denoted s, are drawn from the training time series and a similarity search is conducted by looking up whether similar candidates have been previously considered (lines 4-8). Equation 3 formalizes the procedure as a similarity search over a list $\mathcal{L}$ (e.g., $\mathcal{A}$ or $\mathcal{R}$), considering candidates having same length ($length()$). Please note that in the concrete implementation we use a pruning of the Euclidean distance computations, by stopping comparisons exceeding the threshold ϵ.

$$\textbf{LookUp}(s, \mathcal{L}, \epsilon) \;\; := \;\; \exists q \in \mathcal{L} \;\; | \;\; ||s-q||^2 < \epsilon \, \wedge \, |s| = |q| \qquad (3)$$

3.3.3 Incremental Nearest Neighbor Distances

In case a candidate is found to be novel (not similar to previously considered), then the distance of the candidate to training series are computed using Equation 1 and stored as d^s. Our approach evaluates the **joint** accuracy of accepted shapelets, so far, using a nearest neighbor classifier over the shapelet-transformed data, i.e. distances of series to accepted shapelets.

When checking how does a new ($|\mathcal{A}|$+1)-st candidate influence the accuracy of $|\mathcal{A}|$ currently accepted candidates, an important speed-up trick can be used. We can pre-compute the distances among shapelet-transformed features in an incremental fashion. The distances among series in the shapelet-transformed space are stored in a distance matrix, denoted X, as shown in Equation 4.

$$X_{i,m}\left(D\right) = \sum_{j=1}^{|\mathcal{A}|} \left(D_{i,j} - D_{m,j}\right)^2, \ \ \forall i \in \{1,\dots,N\}, \forall m \in \{1,\dots,N\} \quad (4)$$

We propose a novel trick, which can add the distance contribution of a new candidate to the distance matrix in an incremental manner. When adding one more attribute d^s to the shapelet-transformed data D, we can use the previously computed pair-wise distances to incrementally update the new pair-wise distances as shown in Equation 5.

$$\begin{aligned} X_{i,m}\left(D \, \cup \, d^s\right) &= \sum_{j=1}^{|\mathcal{A}|} \left(D_{i,j} - D_{m,j}\right)^2 + \left(d_i^s - d_m^s\right)^2 \\ &= X_{i,m}\left(D\right) + \left(d_i^s - d_m^s\right)^2 \end{aligned} \quad (5)$$

Those steps correspond to lines 10-12 and 19-21 in Algorithm 2. It is trivial to verify that this technique can improve the runtime of a nearest neighbor from $\mathcal{O}\left(N^2|\mathcal{A}|\right)$ to $\mathcal{O}\left(N^2\right)$, which means that we can avoid recomputing distances among previously accepted $|\mathcal{A}|$-many shapelets, yielding a speed-up factor $|\mathcal{A}|$ for every *considered* shapelet candidate.

3.3.4 Supervised Shapelet Selection

In case the contribution of a unique candidate improves the classification accuracy of a nearest neighbor classifier, then the shapelet is added to the accepted list and the distance vector is stored in a shapelet-transformed data representation $\mathcal{D}$, in order to be later on used for classifying the test instances. Otherwise, the shapelet is inserted to the rejected list and the contribution of the candidate to the distance matrix X is rolled back. The classification accuracy of the distances between series and a set of shapelets is measured by the nearest neighbor accuracy of the cumulative distance matrix X. The accuracy over the training data is formalized in Equation 6.

$$\textbf{Accuracy}(X,Y) := \frac{1}{N}\left|\left\{ i \in \{1,\dots,N\} \ | \ Y_i = Y_{\text{argmin}_{m,m\neq i} X_{i,m}} \right\}\right| \quad (6)$$

The mechanism described in Section 3.3.3 and Section 3.3.4 consists of a supervised variable selection for shapelet-transformed features (Guyon and Elisseeff, 2003). The strategy is a "Forward greedy selection" where shapelets are *Accepted* incrementally if they improve the accuracy (Guyon and Elisseeff, 2003).

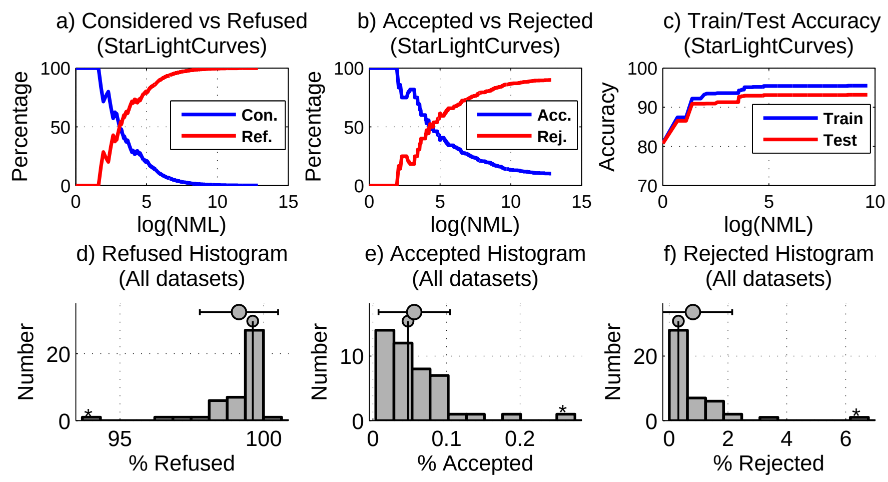


Fig. 3 **a,b,c)** Relations of refused, rejected and accepted candidate shapelets, and the resulting accuracy, for the Starlight dataset; **d,e,f)** Histograms of refused, accepted and rejected candidate percentages over all 45 UCR datasets.

3.3.5 Number of Sampled Candidates

Algorithm 2 samples shapelet candidates randomly, however the total number of sampled candidates is NQR, that upper bounds the total possible series segments of a dataset. Our method could perform competitively even if we would sample a subset of the total possible candidates, as indicated by Figure 3 plot c). That plot illustrates that the train and test accuracy on the StarLightCurves dataset converges well before trying out all the candidates. However, since the state of the art methods try out all the series segments as candidates, we also opted for the same approach. In that way, the runtime comparison against the baselines provides an isolated hint on the impact of the pruning strategy.

3.3.6 An Illustration of The Process

We present the main idea of our method with the aid of Figure 3. Sub-figures *a), b), c)* display the progress of the method on the StarLightCurves dataset, the largest dataset from the UCR collection (Keogh et al., 2011). The fraction of considered (accepted+rejected) shapelets are shown in *a)* with respect to the total candidates in the X-axis. As can be seen, the first few candidates are considered until the accepted and rejected lists are populated with patterns from the dataset. Afterwards, the algorithm starts refusing (pruning/not considering) previously considered candidates within the 25-th percentile threshold, while in the end, an impressive 99.97% of candidates are pruned. In fact this behavior is not special to the StarLightCurves dataset. We ran the algorithm over all the 45 datasets of the UCR collection and measured the fraction of refused candidates as displayed in the histogram of sub-figure *d)*. In average, 99.14% of candidates can be pruned, with cross-validated values p, r on the training data for each dataset.

Among the considered candidates, a supervised selection of shapelets is carried on by accepting only those candidates that improve the classification accuracy. Sub-figure *b)* shows that the number of rejections overcomes the number of acceptances as candidates are evaluated, which validates the current belief that very few shapelets can accurately classify a dataset (Ye and Keogh, 2009). As a consequence of the accepted shapelets, the train and test accuracy of the method on the dataset is improved as testified by sub-figure *c)*. With respect to all datasets of the UCR collection, histograms of sub-figures *d), e)* show that on average **only** 0.06% of candidates are accepted and 0.81% are rejected.

3.3.7 A further intuition

The similarity based pruning of candidates can be compared to a particular type of clustering where the considered candidates represent centroids. In principle, the mechanism resembles fast online clustering methods (Allan, Papka and Lavrenko, 1998). Figure 4 illustrates how the considered shapelets (blue) can be perceived as an ϵ threshold clustering of the refused candidates (gray). Each cluster is represented by a hyper-ball of radius ϵ in a Φ_r-dimensional space, for Φ_r being the shapelet length. For the sake of illustration we selected random points of the shapelets and printed 2-dimensional plots of the 6 considered candidates and 7036 refused candidates from the MALLAT dataset.

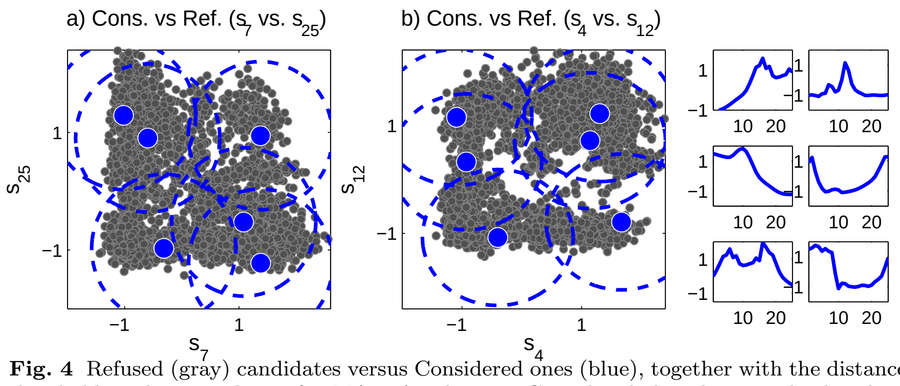


Fig. 4 Refused (gray) candidates versus Considered ones (blue), together with the distance threshold circles, are shown for MALLAT dataset. Considered shapelets are displayed on the right. Parameters: $r = 0.125$, $p = 25$, (i.e. radius is $\epsilon = 1.26$), $\Phi_r = 25$.

The threshold distance used for pruning similar candidates has a significant effect on the quantity of refused candidates. Figure 5 shows that an increase of the percentile parameter both deteriorates the classification accuracy (sub-figure *a)*) and significantly shortens the running time (sub-figure *b)*). The higher the distance threshold percentile, the more distant segments will be considered similar and subsequently more candidates will be refused. In order to avoid a severe accuracy deterioration, the percentile parameter p needs to be fixed by cross-validating over the training accuracy.

3.4 Piecewise Aggregate Approximation (PAA)

The Piecewise Aggregate Approximation (PAA) is a dimensionality reduction technique that shortens time series by averaging consecutive values (Chakrabarti, Keogh, Mehrotra and Pazzani, 2002). Algorithm 3 illustrates how the time series of a dataset can be compressed by a ratio r. For instance, if $r = \frac{1}{4}$ then every four consecutive points are replaced by their average values.

Algorithm 3: PiecewiseAggregateApproximation: Compress every series by a ratio r.

```
   Data: Time series data $T \in \mathbb{R}^{N \times Q}$, PAA ratio $r \in \{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots\}$
   Result: $T^{\text{PAA}} \in \mathbb{R}^{N \times \lceil Q\, r \rceil}$
 1 $T \leftarrow \mathbf{0}_{N \times \lceil Q\, r \rceil}$;
 2 for $i = 1, \dots, N,\ j = 1, \dots, \lceil Q\, r \rceil$ do
 3     for $k = \lceil \frac{1}{r}(j-1) + 1 \rceil, \dots, \lceil \frac{j}{r} \rceil$ do
 4         $T^{\text{PAA}}_{i,j} \leftarrow T^{\text{PAA}}_{i,j} + T_{i,k}$;
 5     end
 6     $T^{\text{PAA}}_{i,j} \leftarrow T^{\text{PAA}}_{i,j}\ r$;
 7 end
 8 return $T^{PAA}$
```

PAA significantly reduces the discovery time of shapelets as shown in Figure 6.b for selected datasets. Moreover, subfigure a) shows that the classification accuracy does not deteriorate significantly because time-series data can be compressed without undermining the series pattern.

The exact amount of PAA reduction and the percentile of the pruning similarity threshold are hyper-parameters that need to be fixed per each dataset using the training data. For instance, Figure 6.c illustrates the accuracy heatmap

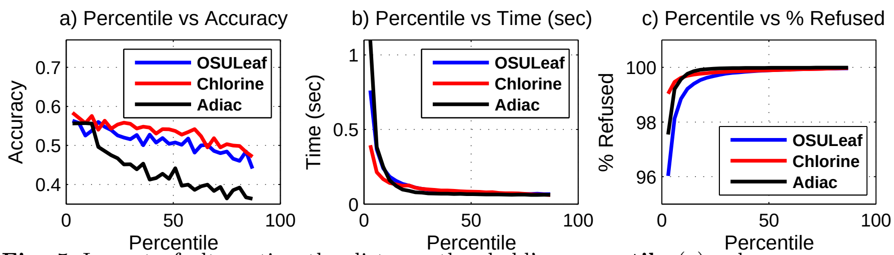


Fig. 5 Impact of alternating the distance threshold's **percentile** (p) value on accuracy, discovery time and the fraction of refused candidates.

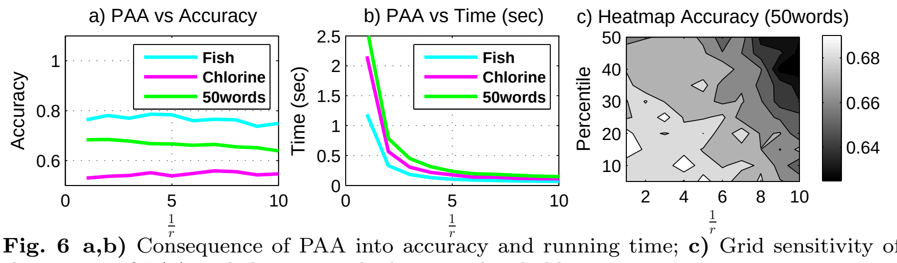


Fig. 6 a,b) Consequence of PAA into accuracy and running time; **c)** Grid sensitivity of the impact of PAA and the percentile distance threshold over accuracy.

on the 50words dataset as a result of alternating both parameters. As shown, the best accuracy is achieved for moderate values of percentile threshold and compression. In contrast, (i) excessive compression and (ii) high threshold percentiles can deteriorate accuracy by (i) destroying informative local patterns by compression and (ii) pruning qualitative variations of shapelet candidates.

3.5 Algorithmic Analysis of the Runtime Speed-Up

The runtime of shapelet discovery algorithms, which explore candidates among series segments, is upper bounded by the number of candidates in a dataset. Given N-many training series of length Q, the total number of shapelet candidates has an order of $\mathcal{O}\left(NQ^2\right)$, while the time needed to find the best shapelet is $\mathcal{O}\left(N^2Q^4\right)$. Please note that the discovery time is quadratic in terms of the number of candidates. Applying Piecewise Aggregate Approximation (PAA), in order to reduce the length of time-series by a ratio $r \in \{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots, ...\}$, does alter the runtime complexity into $\mathcal{O}\left(N^2\left(r\,Q\right)^4\right)$ translated to $\mathcal{O}\left(\mathbf{r^4}N^2Q^4\right)$. In other words, PAA reduces the running time by a factor of $\mathbf{r^4}$. Furthermore, similarity pruning of candidates has a determinant role in reducing the runtime complexity. Let us denote the fraction of considered candidates as $f := \frac{\#\text{accepted}+\#\text{rejected}}{NQ^2}$. Therefore, if executed after a PAA reduction, our algorithm reduces the number of candidates to $\mathcal{O}\left(fN\left(rQ\right)^2\right)$ and impacts the total runtime complexity by $\mathcal{O}\left(fN\left(rQ\right)^2 \times \left(N\left(rQ\right)^2 + 2N^2\right)\right)$, which is upper bounded by $\mathcal{O}\left(fr^4N^2Q^4\right)$, since usually $\left(rQ\right)^2 \gg 2N$. Ultimately, the expected runtime reduction factor achieved by this paper is upper-bounded by $\mathbf{fr^4}$.

There is an additional term that adds up into the runtime complexity: the time needed to check whether any sampled candidate has been previously considered. Such a complexity is $\mathcal{O}\left(N(rQ^2) \times f|r\Phi_*|\right)$, in other words, all candidates times the time needed to search for ϵ similarity on the accepted and rejected lists (f-considered candidates having length $|r\Phi_*|$). Since $|r\Phi_*| \sim \mathcal{O}\left(rQ\right)$, then the whole operation has a final complexity of $\mathcal{O}\left(fr^3NQ^3\right)$. Such a complexity is smaller than the time needed to evaluate the accuracy of the candidates ($\mathcal{O}\left(fr^4N^2Q^4\right)$), therefore does not alter the big-O complexity.

Let us illustrate the theoretically expected speed-up via an example. Assume we compress time-series into a quarter of the original lengths, i.e. $r = \frac{1}{4}$. The average fraction of considered shapelets in the UCR datasets is $f = 0.0086$, as previously displayed in Figure 3. Therefore, a run-time reduction factor of $fr^4 = (0.0086)(0.065) \approx 5.3 \times 10^{-4}$ is expected. As shown, the expected theoretic runtime speedup can be 4 orders of magnitude compared to the exhaustive shapelet discovery. A detailed analysis of the effects of the dimensionality reduction (PAA compression) and pruning on the runtime performance is provided in Section 4.6. Furthermore, in Section 4.3 we will empirically demonstrate that our method is faster than existing shapelet discovery methods.

3.6 Effect Analysis of Supervised Shapelet Selection

In this subsection we analyze the effects of the supervised shapelet selection mechanism. In particular, one could ask whether the incremental Nearest Neighbor (NN) method of Section 3.3.3 is better than not pruning based on accuracy. Stated alternatively, would *accepting* all *considered* candidates (no *rejection* as per the taxonomy of Section 3.3.1) be equally preferable?

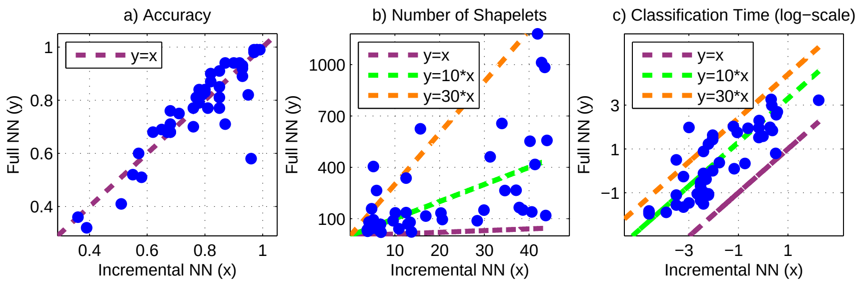


Fig. 7 Comparing an Incremental NN against a Full NN in terms of accuracy, model complexity and classification time on 45 datasets from the UCR collection

There are two primary reasons why an incremental NN is needed: *interpretability* and *classification time*. Meanwhile, Figure 7 helps clarifying both points. One of the motivations for shapelets is interpretability, therefore visual comprehension demands a small set of shapelets (Ye and Keogh, 2009). As is seen in Figure 7.b) a full NN (no *rejected* candidates) ends up having on average 1477% more accepted candidates than our incremental approach. As a form of Variable Subset Selection, our incremental NN is expected to achieve comparable accuracy compared to a NN with a full set of features. As Figure 7.a) indicates, the full NN has slightly higher accuracy values, however the differences are way insignificant according to a Wilcoxon signed rank test indicating a p-value of $p = 0.65272$ with a significance level of $p \leq 0.05$. The last argument in favor of an incremental NN approach is the classification time, which is a trivial consequence of having more features (i.e. more accepted shapelets). Figure 7.c) shows the comparisons of classification times between the two approaches, with the full NN being on average 1566% slower.

4 Experimental Results

4.1 Baselines

In order to evaluate the efficiency of the proposed method Scalable Shapelet Discovery (denoted by **SD**), the fastest state-of-the-art shapelet discovery methods were selected, being:

1. **Logical Shapelet (Mueen et al., 2011) (denoted as LS)**: advances the original shapelet discovery method (Ye and Keogh, 2009) by one order of magnitude, via: (i) caching and reusing computations, and (ii) applying an admissible pruning of the search space (Mueen et al., 2011).
2. **Fast Shapelet (Rakthanmanon and Keogh, 2013) (denoted as FS)**: is a recent state-of-the-art method that proposes a random projection technique on the SAX representation by filtering potential candidates (Rakthanmanon and Keogh, 2013). FS has been shown to reduce the shapelet discovery time of LS by two to three orders of magnitude (Rakthanmanon and Keogh, 2013).
3. **Improved Fast Shapelet (denoted as FS++)**: is a variation of FS that we created for the sake of being fair to the FS baseline. The original FS paper iterates through all the shapelet lengths from one to the length of the series. In comparison, our method SD iterates through a subset of the possible lengths (Φ) as mentioned in Section 4.2. In order to be fair (with respect to runtime), we created a variant of the FS, named FS++, that also iterates through the same subsets of shapelet lengths that SD does.

The comparison against the listed state-of-the-art methods will testify the efficiency of our method in terms of runtime scalability. When proposing a faster solution to a supervised learning task, it is crucial to also demonstrate that the speed-up does not deteriorate the prediction accuracy. For this reason, we payed attention to additionally compare the classification accuracy against the baselines.

4.2 Setup and Reproducibility

In order to demonstrate the speed-up achievements of the proposed shapelet discovery method, we use the popular collection of time-series datasets from the UCR collection (Keogh et al., 2011). The collection includes 45 univariate time-series datasets of different number of instances, different number of classes and lengths, found on (Keogh et al., 2011).

Our Scalable Shapelet Discovery method, denoted as **SD**, requires the tuning of two parameters, the aggregation ratio r and the threshold percentile p. The parameters were searched for each dataset via cross-validation using only the training data. The combination (r, p) that yielded the highest accuracy on the training set was selected. A grid search was conducted with parameter ranges being $r \in \left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}\right\}$ and $p \in \{15, 25, 35\}$. We start with the fastest configuration $r = \frac{1}{8}$ and $p = 35$. Subsequently we increase r and decrease p with the values of the range, one at a time. The selection stops when there is no more increase in accuracy as a result of relaxing the dimensionality reduction r and threshold p. Finally, the winning combination of parameters was applied over the test data. We would like to note that we used three shapelet lengths for all our experiments, i.e. $L = 3$ and $\Phi = \{0.2Q, 0.4Q, 0.6Q\}$. In order to neutralize the randomness effect, all the results of our method represent the averages over five different repetitions.

We used the Java programming language to implement our method (SD), while the other baselines (LS, FS, FS++) are implemented in C++. We decided to use the C++ source codes provided and optimized by the respective baseline paper authors (Rakthanmanon and Keogh, 2013; Mueen et al., 2011), in order to avoid typical allegations on inefficient re-implementations. Finally, we are presenting the exact number of accepted shapelets per each dataset and the respective percentages of the accepted, rejected and refused candidates in the columns merged under "SD Performance". **All** experiments (both our method and the baselines) were conducted in a *Sun Grid Engine* distributed cluster with 40 node processors, each being *Intel Xeon* E5-2670v2 with speed 2.50GHz and 64GB of shared RAM for all nodes. The operating system was *Linux CentOS* 6.3. All the experiments were launched using the same cluster parameters.

The authors are committed to promote experimental reproducibility. For this reason the source code, all the datasets, the executable file and instructions are provided unconditionally[1].

4.3 Highly Qualitative Runtime Results

The empirical results include both the discovery time and the classification accuracy of our method SD against baselines for 45 UCR datasets. Table 2 contains a list of results per dataset, where the discovery time is measured in seconds. A time-out threshold of 24 hours was set for the discovery of shapelets of a single dataset. As can be seen, the Logical Shapelet (LS) exceeded the time-out threshold in a considerable number of datasets. The reader is invited to notice that 24 hours (86400 seconds) is a very large threshold, given that our method SD often finds the shapelets within a fraction of one second, as for instance in the 50words dataset.

It can be clearly deduced that our method SD is faster than the fastest existing baselines LS (Mueen et al., 2011) and FS (Rakthanmanon and Keogh,

[1] `fs.ismll.de/publicspace/ScalableShapelets`

Table 2 **Parameters** of SD and **Runtime Results** of SD and State-of-the-art baselines over 45 UCR datasets (n/a denotes a 24h time-out)

No	**Dataset**	**SD Params.**		**Discovery Time (seconds)**			
		r	**p**	**LS**	**FS**	**FS++**	**SD**
1	50words	0.250	35	n/a	2198.1	35.2	**0.36**
2	Adiac	0.500	15	12683.2	332.6	6.4	**0.25**
3	Beef	0.125	35	242.3	194.9	1.9	**0.03**
4	CBF	0.500	35	66.9	10.9	0.4	**0.03**
5	Chlorine.	0.125	15	36402.3	760.3	13.9	**0.17**
6	CinC_ECG.	0.125	25	2150.0	4398.9	9.9	**0.34**
7	Coffee	0.250	35	621.9	22.5	0.2	**0.03**
8	Cricket_X	0.250	35	n/a	3756.0	47.9	**0.63**
9	Cricket_Y	0.250	35	n/a	3605.7	45.7	**0.52**
10	Cricket_Z	0.250	35	n/a	4679.2	46.2	**0.67**
11	Diatom.	0.125	15	184.3	15.6	0.2	**0.02**
12	ECG200	0.125	15	618.8	16.3	0.9	**0.04**
13	ECGFive.	0.500	15	47.6	3.6	0.1	**0.03**
14	FaceAll	0.500	35	16255.5	757.5	27.0	**1.25**
15	FaceFour	0.500	35	561.2	102.9	1.0	**0.11**
16	FacesUCR	0.500	35	2528.5	280.3	8.7	**0.33**
17	Fish	0.250	25	11153.0	935.6	6.7	**0.16**
18	Gun_Point	0.500	25	266.1	9.5	0.3	**0.04**
19	Haptics	0.500	25	n/a	12491.0	31.1	**1.78**
20	InlineSkate	0.125	15	n/a	22677.2	42.6	**0.61**
21	ItalyPower.	1.000	25	4.9	0.4	0.1	**0.02**
22	Lighting2	0.500	35	5297.6	1131.3	5.0	**1.89**
23	Lighting7	0.500	35	8619.3	322.8	3.7	**0.43**
24	MALLAT	0.125	35	1254.9	1736.5	6.2	**0.08**
25	MedicalImages	0.500	35	19325.2	371.5	8.5	**0.60**
26	MoteStrain	1.000	15	6.9	3.1	0.1	**0.05**
27	Non.Fat.ECG.1	0.250	25	n/a	70970.6	254.2	**7.03**
28	Non.Fat.ECG.2	0.125	25	n/a	50898.0	232.8	**4.99**
29	OliveOil	0.125	15	502.3	107.2	0.8	**0.05**
30	OSULeaf	0.125	25	14186.5	1629.7	20.0	**0.15**
31	Sony.I	1.000	35	4.6	1.1	0.1	**0.02**
32	Sony.II	1.000	35	9.8	1.3	0.1	**0.03**
33	StarLight.	0.125	25	n/a	21473.5	78.5	**3.19**
34	SwedishLeaf	0.500	25	11953.6	451.7	12.9	**0.36**
35	Symbols	0.250	25	894.3	93.0	0.6	**0.04**
36	synthetic.	0.250	35	3667.4	63.9	3.6	**0.07**
37	Trace	0.500	35	4626.9	181.0	1.7	**0.13**
38	Two_Patterns	0.500	35	65783.1	957.2	37.7	**1.71**
39	TwoLeadECG	1.000	25	14.3	1.3	0.03	**0.02**
40	uWave.X	0.250	25	n/a	4827.5	54.1	**4.94**
41	uWave.Y	0.250	25	n/a	4379.6	56.6	**3.69**
42	uWave.Z	0.125	25	n/a	5215.9	50.9	**1.83**
43	wafer	0.500	35	34653.1	190.5	5.0	**1.39**
44	WordsS.	0.250	25	n/a	1140.0	18.7	**0.31**
45	yoga	0.250	15	11389.0	1711.6	11.2	**0.34**
Total Wins				**0**	**0**	**0**	**45**
Average Rank				**0.000**	**0.000**	**0.000**	**1.000**

Table 3 **Parameters** of SD and **Classification Accuracy Results** of SD and SOTA baselines over 45 UCR datasets (n/a denotes a 24h time-out)

No	Dataset	#Acc	Classification Accuracy			
			LS	**FS**	**FS++**	**SD**
1	50words	39	n/a	0.511	0.446	**0.680**
2	Adiac	28	**0.586**	0.574	0.486	0.583
3	Beef	5	**0.567**	0.513	0.503	0.507
4	CBF	5	0.886	0.935	0.907	**0.975**
5	Chlorine.	13	**0.618**	0.579	0.558	0.553
6	CinC_ECG.	13	0.699	0.751	0.656	**0.773**
7	Coffee	4	**0.964**	0.921	0.907	0.961
8	Cricket_X	43	n/a	0.472	0.368	**0.672**
9	Cricket_Y	42	n/a	0.480	0.464	**0.675**
10	Cricket_Z	44	n/a	0.438	0.376	**0.673**
11	Diatom.	4	0.801	0.886	**0.928**	0.896
12	ECG200	10	**0.870**	0.766	0.786	0.818
13	ECGFive.	5	0.994	**0.995**	0.994	0.953
14	FaceAll	40	0.659	0.631	0.571	**0.714**
15	FaceFour	6	0.489	**0.917**	0.881	0.820
16	FacesUCR	31	0.662	0.703	0.654	**0.847**
17	Fish	14	0.777	**0.809**	0.785	0.755
18	Gun_Point	6	0.893	**0.933**	0.915	0.931
19	Haptics	13	n/a	**0.376**	0.347	0.356
20	InlineSkate	13	n/a	0.266	0.282	**0.385**
21	ItalyPower.	6	**0.936**	0.877	0.796	0.920
22	Lighting2	9	0.426	0.707	0.698	**0.795**
23	Lighting7	16	0.548	0.630	0.485	**0.652**
24	MALLAT	7	0.656	**0.939**	0.926	0.926
25	MedicalImages	34	0.587	0.596	0.494	**0.676**
26	MoteStrain	5	**0.832**	0.783	0.767	0.783
27	Non.Fat.ECG.1	41	n/a	0.766	0.622	**0.814**
28	Non.Fat.ECG.2	44	n/a	0.802	0.635	**0.855**
29	OliveOil	5	**0.833**	0.723	0.773	0.790
30	OSULeaf	21	**0.686**	0.680	0.555	0.566
31	Sony.I	4	**0.860**	0.686	0.802	0.850
32	Sony.II	5	0.846	0.792	**0.945**	0.780
33	StarLight.	20	n/a	**0.942**	0.932	0.933
34	SwedishLeaf	30	0.813	0.779	0.725	**0.849**
35	Symbols	4	0.643	**0.933**	0.756	0.865
36	synthetic.	11	0.470	0.922	0.870	**0.983**
37	Trace	7	**1.000**	0.994	0.999	0.965
38	Two_Patterns	38	0.539	0.310	0.753	**0.981**
39	TwoLeadECG	4	0.856	**0.928**	0.798	0.867
40	uWave.X	44	n/a	0.707	0.580	**0.761**
41	uWave.Y	41	n/a	0.608	0.466	**0.671**
42	uWave.Z	37	n/a	0.627	0.565	**0.676**
43	wafer	10	**0.999**	0.998	0.949	0.993
44	WordsS.	35	n/a	0.437	0.389	**0.625**
45	yoga	17	**0.740**	0.705	0.697	0.625
Total Wins			**13**	**9**	**2**	**21**
Average Rank			**2.313**	**2.178**	**3.089**	**1.889**

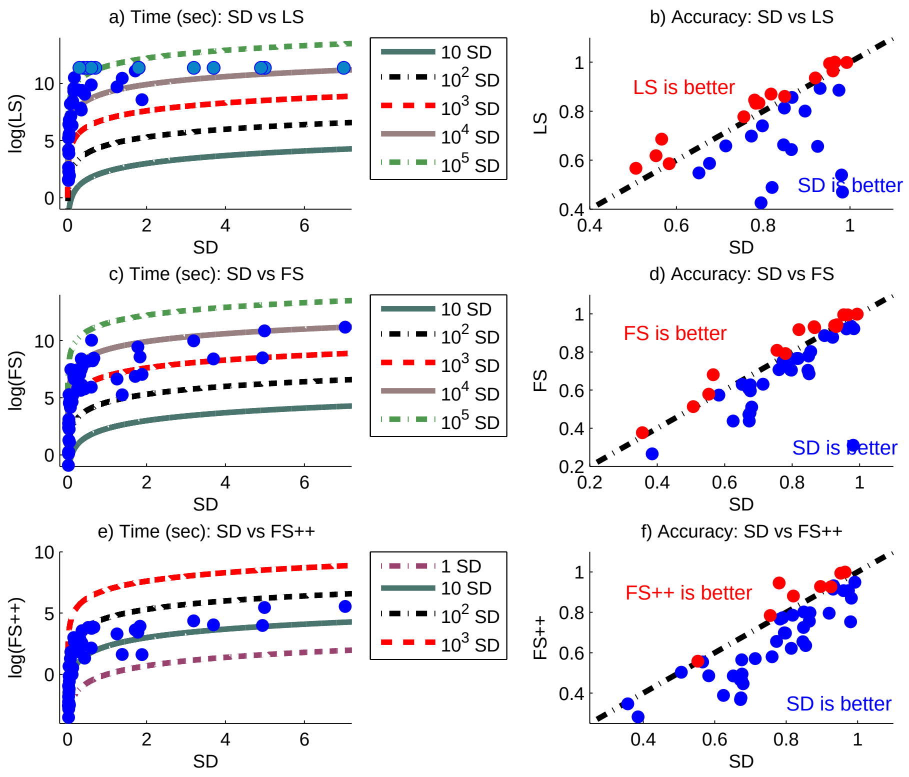


Fig. 8 Time and accuracy comparison of our method (denoted **SD**) against state-of-the-art methods both in terms of discovery time and classification accuracy for all the 45 UCR datasets.

2013). There is no dataset where any of the baselines is faster. Even, our modification of FS, i.e. the FS++, is considerably slower than SD. For instance, it took only 3.19 seconds for our method to find the shapelets of the StarLightCurves dataset, which has 1000 training instances each having 1024 points. The high-level conclusion from the discovery time results is: "Since the introduction of shapelets in 2009, time-series community believed shapelets are very useful classification patterns, but finding them is slow. This paper demonstrates that shapelets can be discovered very fast."

The discovery time measurements do not include the time needed by a practitioner to tune the parameters of the methods. While our method has two parameters (p and r, totaling $3 \times 4 = 12$ combinations, see Section 4.2), the strongest baseline (Fast Shapelet) has more parameters, concretely four: the reduced dimensionality and cardinality of SAX, the random projection iterations and the number of SAX candidates (denoted d,c,r,k in the original paper (Rakthanmanon and Keogh, 2013)).

4.4 Competitive Prediction Accuracy

In addition, our results are atypical in another positive aspect. Most scalability papers propose speed-ups of the learning time by sacrificing a certain fraction of the prediction accuracy. In contrast, our results show that our method is both faster and more accurate than the baselines. The winning method that achieves the highest accuracy on each dataset (on each row) is distinguished in bold. Our method has more wins than the baselines (21 wins against 13 of the second best method) and also a better rank (1.889 against 2.178 of the second best method). The accuracy improvement arises from the joint interaction of accepted shapelets as predictors (distance matrix X in Algorithm 2), while the baselines measure the quality of each shapelet separately, without considering their interactions during the discovery phase (Ye and Keogh, 2009; Mueen et al., 2011; Rakthanmanon and Keogh, 2013). Incorporating the interactions among shapelets into the prediction model has been recently shown to achieve high classification accuracy (Grabocka, Schilling, Wistuba and Schmidt-Thieme, 2014).

4.5 Speed-Up Analysis

In order to show the speed-up factor of our method with respect to the (former) state-of-the-art, we provide another presentation of the results in Figure 8. The three plots on the left side show the discovery time of SD in x-axis and the logarithm of the discovery time of each baseline as the y-axis. As can be easily observed from the illustrative order lines, SD is 4 to 5 orders of magnitude faster than the Logical Shapelet (LS) and 3 to 4 orders of magnitude faster than the Fast Shapelet (FS). The datasets where LS exceeds the 24 hour threshold are depicted in light blue. In addition, FS++ is faster than FS because it iterates over less shapelet length sizes, yet it is still 1 to 2 orders of magnitude slower than SD.

The plots on the right represent scatter plots of the classification accuracy of SD against the baselines. While generally better than LS and FS, our method SD is largely superior to FS++. Such a finding indicates that the accuracy of the Fast Shapelet (FS) is dependent on trying shapelet candidates from a fine-grained set of lengths, while our method is very accurate even though it iterates over few shapelet lengths.

4.6 A Modular Decomposition of the Performance

We have already seen that our proposed method, SD, outperforms significantly the state-of-the-art in terms of runtime and produces even better prediction accuracy. Nevertheless, there are a couple of questions that can be addressed to our method, such as:

Table 4 Modular Decomposition of The Performance of Our Method (SD), N/A Denotes a 24H time-out

Dataset	**Discovery Time (seconds)**				**Classification Accuracy**			
	✗PAA	**✓PAA**	**✗PAA**	**✓PAA**	**✗PAA**	**✓PAA**	**✗PAA**	**✓PAA**
	✗prun.	**✗prun.**	**✓prun.**	**✓prun.**	**✗prun.**	**✗prun.**	**✓prun.**	**✓prun.**
50words	4028.85	154.74	5.24	0.36	0.684	**0.701**	0.679	0.680
Adiac	799.06	153.21	0.89	0.25	**0.624**	0.555	0.604	0.583
Beef	61.35	0.65	0.54	0.03	0.533	**0.600**	0.500	0.507
CBF	2.22	0.57	0.37	0.03	**0.992**	0.964	0.929	0.975
Chlorine.	1598.05	30.14	2.40	0.17	0.527	**0.596**	0.539	0.553
CinC_ECG.	3718.48	11.74	12.71	0.34	**0.809**	0.768	0.776	0.773
Coffee	11.79	1.27	0.39	0.03	**0.964**	0.893	0.893	0.961
Cricket_X	4218.58	141.80	23.63	0.63	**0.697**	**0.697**	0.669	0.672
Cricket_Y	3953.86	137.75	14.20	0.52	**0.715**	0.687	0.677	0.675
Cricket_Z	5313.96	132.06	40.17	0.67	0.700	0.682	**0.726**	0.673
Diatom.	5.00	0.50	0.50	0.02	0.915	**0.948**	0.827	0.896
ECG200	14.59	0.70	0.42	0.04	0.820	0.800	**0.830**	0.818
ECGFive.	1.41	0.40	0.36	0.03	**0.999**	0.945	0.981	0.953
FaceAll	1276.24	297.23	4.87	1.25	0.720	**0.731**	0.724	0.714
FaceFour	18.04	3.10	1.21	0.11	0.852	0.898	**0.943**	0.820
FacesUCR	107.35	24.74	2.70	0.33	**0.871**	0.868	0.841	0.847
Fish	1808.85	46.46	1.61	0.16	0.817	**0.846**	0.800	0.755
Gun_Point	7.69	1.55	0.60	0.04	0.900	0.913	**0.953**	0.931
Haptics	17273.44	2634.99	6.59	1.78	0.354	**0.373**	0.321	0.356
InlineSkate	34776.14	99.61	19.82	0.61	**0.411**	0.342	0.313	0.385
ItalyPower.	0.77	0.49	0.62	0.02	**0.936**	0.925	0.915	0.920
Lighting2	843.42	90.84	12.23	1.89	**0.852**	0.836	0.836	0.795
Lighting7	120.39	20.15	4.89	0.43	0.699	**0.740**	0.685	0.652
MALLAT	2295.97	6.25	1.99	0.08	0.909	0.938	**0.941**	0.926
MedicalI.	349.15	57.75	1.76	0.60	0.625	0.658	0.668	**0.676**
MoteStrain	0.91	0.62	0.21	0.05	0.734	**0.815**	0.777	0.783
Non.ECG.1	n/a	35833.59	36.79	7.03	n/a	**0.840**	0.795	0.814
Non.ECG.2	n/a	11086.13	58.18	4.99	n/a	0.852	**0.858**	0.855
OliveOil	75.17	0.90	1.12	0.05	**0.900**	0.800	0.700	0.790
OSULeaf	2379.27	16.64	3.18	0.15	0.570	0.541	**0.583**	0.566
Sony.I	1.28	0.62	0.76	0.02	0.829	**0.902**	0.792	0.850
Sony.II	0.46	0.47	0.86	0.03	0.727	0.774	0.742	**0.780**
StarLight.	n/a	4673.16	74.14	3.19	n/a	**0.933**	0.929	**0.933**
SwedishLeaf	830.60	301.63	1.24	0.36	**0.869**	0.856	0.856	0.849
Symbols	27.58	1.64	0.58	0.04	0.805	0.787	0.819	**0.865**
synthetic.	51.03	6.01	0.56	0.07	0.980	**0.993**	0.980	0.983
Trace	138.09	25.15	0.60	0.13	0.950	**0.990**	0.960	0.965
Two_Patterns	4572.63	1216.45	2.78	1.71	0.985	0.984	**0.986**	0.981
TwoLead.	0.54	0.88	0.41	0.02	**0.932**	0.774	**0.932**	0.867
uWave.X	27142.53	1565.73	19.46	4.94	0.757	0.745	**0.762**	0.761
uWave.Y	25276.28	1385.23	16.74	3.69	0.647	0.643	**0.671**	**0.671**
uWave.Z	24532.05	513.11	14.09	1.83	0.662	0.668	**0.681**	0.676
wafer	6352.87	1750.96	3.31	1.39	0.994	0.994	**0.995**	0.993
WordsS.	1220.31	44.25	3.13	0.31	0.627	**0.639**	0.607	0.625
yoga	5098.73	254.54	3.05	0.34	**0.812**	0.802	0.799	0.625
Total Wins					**14.0**	**15.0**	**12.0**	**4.0**
Average Ranks					**2.2**	**2.3**	**2.5**	**2.7**

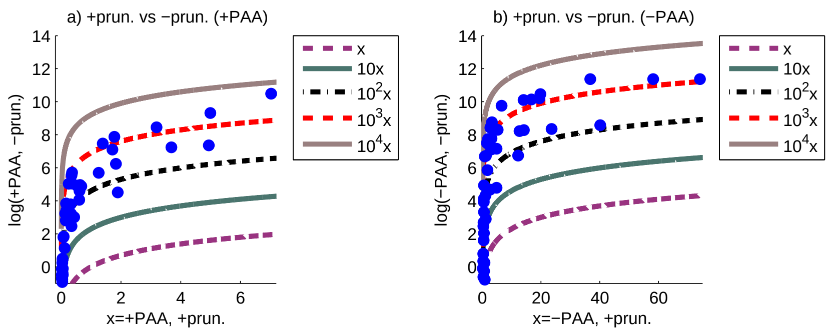


Fig. 9 Runtime comparison (seconds) plots among variants of SD with and without pruning

1. What fraction of SD's runtime reduction is attributed to the novel candidate pruning and what fraction to the PAA compression?
2. To what extent does pruning deteriorate the prediction accuracy?

In order to address those analytic questions we will decompose our method in a modular fashion. Our method, SD, conducts both a PAA approximation and a pruning by the parameters r, p provided in Table 2. In order to isolate the effect of compression and pruning we are creating four variants of our method, namely all the permutations "With/Without PAA compression" and "With/Without Pruning" (w.r.t. to p, r from Table 2). All the decomposed results of the SD variants are shown in Table 4. Note that "No pruning" means $p = 0$, while "no PAA" means $r = 1$. The variant with both pruning and PAA is the same as SD from Section 4.3, which already was shown to be superior to the state of the art.

Looking into the results of Table 4, it is important to observe that the variant with PAA compression alone is significantly faster than the variant without compression (columns 4 vs column 3). However, using pruning without compression is much faster than the exhaustive approach and also much faster than compression alone (column 5 vs. columns 3,4). When pruning and compression are combined (column 6), then the runtime reduction effect multiplies. More concretely, Figure 9 analyses the runtime reduction of SD variants: that use pruning (X-axis) against variants without pruning (Y-axis) for both scenarios with PAA (plot a)) or without PAA (plot b)) compression. As can be clearly deduced, pruning alone has a significant effect on the runtime reduction by 3 to 4 orders of magnitude, compared to the cases where no pruning is employed. While PAA helps our method to be even faster, it is clear that the lion's share of the speedup arises from the proposed pruning mechanism.

There is still a concern on how does pruning affect the classification accuracy. The prediction accuracy results are demonstrated in Table 4 for all the datasets, with the winning variant emphasized in bold. The total wins and the ranks of the variants indicate that the best prediction performance is attributed to the exhaustive methods (no pruning, columns 7,8). Such a finding is natural because exhaustive approaches consider all the candidate variants and can extracts more qualitative minimum distance features. Yet, are the results of the exhaustive variants better with a **statistical significance** margin? Table 5 illustrates the p-values of a Wilcoxon Signed Rank test of statistical significance, for a two-tailed hypothesis with a significance level of 5% ($\alpha = 0.05$).

Table 5 Wilcoxon Statistical Significance Test: p-values (Significance Level 5%, Two-Tailed Hypothesis)

	✘ PAA ✘ prun.	✔ PAA ✘ prun.	✘ PAA ✔ prun.	✔ PAA ✔ prun.
✘ PAA, ✘ prun.	-	0.904	**0.119**	0.046
✔ PAA, ✘ prun.	0.904	-	0.153	**0.112**
✘ PAA, ✔ prun.	**0.119**	0.153	-	0.873
✔ PAA, ✔ prun.	0.046	**0.112**	0.873	-

The p-values which compare variants that use pruning against variants that do not use pruning are shown in bold and correspond to $p = 0.119, p = 0.112$. Therefore, the prediction quality using pruning is not significantly (significance means $p < 0.05$) worse than the exhaustive approach. The final message of this section is: "Pruning of candidates provides 3 to 4 orders of runtime speedup without any statistically significant deterioration in terms of classification accuracy.".

4.7 Comparison To Other State-of-the-art Shapelet Discovery Methods

One would categorize the methods focusing on shapelet discovery into "speed-oriented" and "accuracy-oriented" approaches. The method proposed in this paper SD and the baselines LS, FS were focused on reducing the runtime of shapelet discovery. On the other hand, there are other methods which prioritize on achieving the highest classification accuracy. The most prominent methods on accurate shapelet discovery are "Shapelet Transformation" (Hills et al., 2013) (denoted as ST) and the recently more accurate method "Learning Time-Series Shapelet" (Grabocka et al., 2014) (denoted as LTS).

Table 6 Comparison of the proposed method SD against LTS and ST

Dataset	**Accuracies**			**Running Times (sec)**		
	LTS	**ST**	**SD**	**LTS**	**ST**	**SD**
StarLightCurves	0.964	-	0.933	65657.07	1728000.00*	3.19
Non.Fat.ECG.1	0.865	-	0.814	1448862.51	1728000.00*	7.03
Non.Fat.ECG.2	0.897	-	0.855	1528267.41	1728000.00*	4.99

In this section we aim at showing that while those methods are more accurate, their runtime is much slower than the proposed method SD. For this reason we selected the three largest datasets of the UCR collection as shown in Table 6 and ran SD, ST and LTS on those datasets. In order to be fair to the baselines, LTS and ST were also run on a subset of shapelet lengths $\{0.2Q, 0.4Q, 0.6Q\}$. For both LTS and ST we used the source code provided by the authors. Since those methods are known to be slow we violated the 24 hours time-out of Section 4.2 and instead gave the methods a very large time-out deadline of 20 days to complete the execution over the three datasets. The results of Table 6 indicate that LTS is more accurate than SD in all the dataset, however it took LTS from 18.2 hours to 17.7 days to compute. Furthermore, ST could not finish learning on any of the three datasets within 20 days (time-out denoted by *). On the other hand, SD needs 3.19 to 7.03 seconds to compute the shapelets of those datasets, for a speed-up of up to 346293 times faster. On the other hand, the deterioration in accuracy varies only between 3.3% and 6.2% worse than LTS.

5 Extension to Multivariate Time Series

Multivariate time series has become increasingly popular in the data mining research community. Part of the popularity is attributed to the widespread of affordable motion sensor devices. In fact, multivariate (*synonym:* multidimensional) time series are a generalization of univariate series. In the multivariate case a single time-series instance is composed of different streams measured at the same time. An example of multivariate series are recordings of wearable body sensors, where signal measuring devices are positioned at different parts of the body (Banos, Garcia, Holgado-Terriza, Damas, Pomares, Rojas, Saez and Villalonga, 2014; Banos, Toth, Damas, Pomares and Rojas, 2014).

We can formalize a time-series dataset having N instances and V many dimensions as $T \in \mathbb{R}^{N \times V \times Q*}$, where each series has a different length $T_{i,:,:} \in \mathbb{R}^{Q_i}, \forall i \in \{1, \ldots, N\}$. Whilst the lengths of different instances vary, we assume that the different dimensions within one instance have the same length. In this section we will demonstrate that it is trivial to extend the method proposed in this paper to multivariate time-series datasets. All is needed is to sample shapelet candidates from random dimensions and accept them based on their joint accuracy.

5.1 Addressing Challenges of Multivariate Series

Different series lengths is a common reality for multivariate series. The time-series research community has worked extensively on the UCR collection of datasets, where the time-series instances were preprocessed to have the same lengths. In reality this is rarely the case, however different lengths pose no concrete problem for shapelet-based methods. The minimum distance between a candidate and various series segments is independent on the number of segments. There is however a small problem, the case when a series is shorter than a shapelet. In order to overcome this concern we propose to slid the series over the shapelet, i.e. to measure the minimum distance of a series to all the segments of the shapelet candidate. Equation 7 formalizes the distance between the v-th dimension of the i-th series $T_{i,v,:}$ and a shapelet candidate s.

$$\mathcal{D}(s, T_{i,v,:}) := \begin{cases} \min\limits_{j=1,\ldots,|T_{i,v,:}|-|s|+1} \left\|T_{i,v,j:j+|s|-1} - s\right\|^2 & |T_{i,v,:}| \geq |s| \\ \min\limits_{j=1,\ldots,|s|-|T_{i,v,:}|+1} \left\|s_{j:j+|T_{i,v,:}|-1} - T\right\|^2 & |s| > |T_{i,v,:}| \end{cases} \quad (7)$$

Features from different dimensions are known to improve the classification accuracy, however related works take diverse approaches in how series of different dimensions are incorporated. In terms of shapelets, an early approach extended the concept of univariate shapelets into multi-variate shapelets (Ghalwash and Obradovic, 2012). However, a label might not be associated with certain dimensions or there might be shifts of the starting time of a pattern across dimensions. As a result, a recent work (Cetin, Mueen and Calhoun, 2015) proposed to learn a shapelet-based classifier on each dimension and use a majority voting over the predictions of the per-dimension models.

In contrast, we propose a simple and novel technique to incorporate features from different dimensions. The principle relies on sampling random candidates from random dimensions. Roughly speaking we will harvest accepted and rejected candidates per each dimensions. Distance features from each dimensions will be **jointly integrated** into the same incremental nearest neighbor and filtered by classification accuracy. This mechanism will allow to fuse features of candidates from different dimensions into a joint feature set.

5.2 Algorithm for Multivariate Shapelet Discovery

The concrete implementation of our multivariate method is described in Algorithm 4. Our method selects $NMLV$-many random candidates, from random series i, random dimension v, random length Φ_r and starting at random time index j (lines 4-8). Each random candidate is looked up for similarity to previously considered (accepted or rejected) candidates within that dimension (line 9). If a shapelet candidate is not found to be similar to previous candidates, then its feature vector is computed (line 10) and the pair-wise distance matrix X is updated (line 11-13). In case the candidate improves the overall accuracy (line 14), then it is accepted (lines 15-17) otherwise it is rejected (lines 19-23). In the end we are going to have lists of accepted shapelets for each dimension, such that the features of those accepted shapelets achieve the highest classification accuracy.

5.3 Experimental Results

In order to test our method we compared against the most recent and relevant method which elaborates shapelets for multivariate classification (Cetin et al.,

Algorithm 4: DiscoverShapeletsMultivariate: Scalable discovery of shapelets from multivariate series

```
Data: Multivariate time-series data $T \in \mathbb{R}^{N \times V \times Q*}$, Labels $Y \in \mathbb{N}^N$ Distance,
      Threshold Percentile $p \in [1, \dots, 100]$, Piecewise Aggregate Approximation
      ratio: $r \in \{1, \frac{1}{2}, \frac{1}{4}, \dots\}$, Shapelet lengths: $\Phi \in \mathbb{N}^R$
Result: Accepted shapelets list $\mathcal{A} \in \mathbb{R}^{V \times * \times *}$, Minimum Distances $D \in \mathbb{R}^{* \times *}$
1  $\epsilon_v \leftarrow$ ComputeThreshold($T_{:,v,:}, p, \Phi$), $v \in \{1, \dots, V\}$;
2  $\mathcal{A} \leftarrow \emptyset^V, \mathcal{R} \leftarrow \emptyset^V, D \leftarrow \emptyset, X \leftarrow \mathbf{0}_{N \times N}$, prevAccuracy $\leftarrow -\infty$;
3  for $1, \dots, NMLV$ do
4      Draw random series: $i \sim \mathcal{U}\{1, \dots, N\}$;
5      Draw random dimension: $v \sim \mathcal{U}\{1, \dots, V\}$;
6      Draw random shapelet length: $\Phi_r \sim \mathcal{U}\{\Phi_1, \dots, \Phi_R\}$;
7      Draw random segment start: $j \sim \mathcal{U}\{1, \dots, Q_i - \Phi_r + 1\}$;
8      Selected random candidate: $s \leftarrow T_{i,v,j:j+\Phi_r-1}$;
9      if $\neg$LookUp($s, \mathcal{A}_v, \epsilon_v$) $\wedge$ $\neg$LookUp($s, \mathcal{R}_v, \epsilon_v$) then
10         $d^s \leftarrow$ MinDist($s, T_{:,v,:}$) ;
11         for $i = 1, \dots, N;\ m = i+1, \dots, N$ do
12             $X_{i,m} \leftarrow X_{i,m} + \left(d^s_i - d^s_m\right)^2$;
13         end
14         $\alpha \leftarrow$ Accuracy($X, Y$);
15         if $\alpha >$ prevAccuracy then
16             $\mathcal{A}_v \leftarrow \mathcal{A}_v \cup \{s\}$;
17             $D \leftarrow D \cup \{d^s\}$;
18             prevAccuracy $\leftarrow \alpha$;
19         else
20             $\mathcal{R}_v \leftarrow \mathcal{R}_v \cup \{s\}$;
21             for $i = 1, \dots, N;\ m = i+1, \dots, N$ do
22                 $X_{i,m} \leftarrow X_{i,m} - \left(d^s_i - d^s_m\right)^2$;
23             end
24         end
25     end
26 end
27 return $\mathcal{A}, D$
```

2015). Furthermore, we are going to experiment on four multivariate datasets, whose statistics are displayed in Table 7. Three of them ('HMP', 'M-Health', 'REALDISP') are related to human action recognition using wearable sensors, while 'Characters' represents pen tip trajectories of handwritten characters. The instances of all datasets were randomly divided into train and test sets. It is interesting to note that those datasets are diverse in terms of number of dimensions (V from 3 to 117), number of instances (63 to 1429), number of classes ('Cls.' from 12 to 33) and lengths (109 to 5643).

Another aspect worth consideration is the size of the datasets. For instance REALDISP has a training set size of 889 mb and a total size of 1.75 gb, which is considerably large for labeled time-series data. As a comparison, the largest univariate dataset from the UCR collection is 'StarLightCurves', which has a train set of 16 mb and a total size of 144 mb. In order to address this size challenge we opted for the single fastest parameter configuration for all datasets, with respect to the ranges of Section 4.2, concretely

Table 7 Results of Scalable Shapelet Discovery on Multivariate Datasets, n/a denotes a 24h timeout

Dataset	Data Statistics					Accuracy		Runtime (sec)	
	Train/Te.	V	**Cls.**	**Length**	**Size**	**LS**	**SD**	**LS**	**SD**
Characters	1429/1429	3	20	109-205	8.9mb	0.652	**0.980**	984.55	**3.05**
HMP	487/492	3	21	125-9318	11.5mb	0.474	**0.707**	5362.84	**2.87**
M-Health	63/63	23	12	513-3431	60.3mb	0.746	**0.813**	33781.44	**13.54**
REALDISP	749/749	117	33	318-5643	1.75gb	n/a	**0.723**	n/a	**289.40**

$r = \frac{1}{8}$, $p = 35$. In order to aggregate the randomness effect we present the average figures of five different executions of our method. The runtimes of Table 7 include the classification time. We would like to point out that dataset Characters is retrieved from (Williams, Toussaint and Storkey, 2006), HMP from (Bruno, Mastrogiovanni, Sgorbissa, Vernazza and Zaccaria, 2013), M-Health from (Banos, Garcia, Holgado-Terriza, Damas, Pomares, Rojas, Saez and Villalonga, 2014), while REALDISP retrieved from (Banos, Toth, Damas, Pomares and Rojas, 2014).

The results of Table 7 indicate great achievements in terms of runtime. **Our method can classify the mb-scale datasets in matters of seconds and the gb-scale dataset in matter of minutes.** Concretely, our method needs less than 5 minutes to classify the 1.75gb dataset. Compared to the baseline method (Cetin et al., 2015), our method is 322.8X to 2494.93X times faster on the mb-scale datasets. Unfortunately, the baseline could not complete on the gb-scale dataset under the 24h timeout specified in Section 4.3. On the other hand, our method is more accurate than the baseline on all the datasets. We believe that such a superiority comes from the joint interactions of features from different dimensions, as opposed to learning isolated per-dimension classifiers.

6 Conclusion

Shapelets represent discriminative segments of a time-series dataset and the distances of time-series to shapelets are shown to be successful features for classification. The discovery of shapelets is currently conducted by trying out candidates from the segments (sub-sequences) of the time-series. Since the number of candidate segments is large, the time-series community has spent efforts on speeding up the discovery time of shapelets. This paper proposed a novel method that prunes the candidates based on a distance threshold to previously considered other similar candidates. In a joint fashion, a novel supervised selection filters those shapelets that boost classification accuracy. We empirically showed that our method is 3-4 orders of magnitude faster than the fastest existing shapelet discovery methods, while providing a better prediction accuracy. In addition, we extended our method to multivariate datasets. Results indicate that our approach is are able to classify mb-scale datasets in a matter of seconds and gb-datasets in a matter of minutes, therefore transporting shapelet discovery to the *Big Data* era.

Acknowledgment

This study was partially co-funded by the Seventh Framework Programme (FP7) of the European Commission, through project REDUCTION[2] (#288254).

References

Allan, J., Papka, R. and Lavrenko, V. (1998), On-line new event detection and tracking, *in* 'Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '98, ACM, New York, NY, USA, pp. 37–45.

Banos, O., Garcia, R., Holgado-Terriza, J., Damas, M., Pomares, H., Rojas, I., Saez, A. and Villalonga, C. (2014), mhealthdroid: A novel framework for agile development of mobile health applications, *in* L. Pecchia, L. Chen, C. Nugent and J. Bravo, eds, 'Ambient Assisted Living and Daily Activities', Vol. 8868 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 91–98.

Banos, O., Toth, M. A., Damas, M., Pomares, H. and Rojas, I. (2014), 'Dealing with the effects of sensor displacement in wearable activity recognition', *Sensors* **14**(6), 9995–10023.

Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T. and Zaccaria, R. (2013), Analysis of human behavior recognition algorithms based on acceleration data, *in* 'Robotics and Automation (ICRA), 2013 IEEE International Conference on', pp. 1602–1607.

Cetin, M. S., Mueen, A. and Calhoun, V. D. (2015), Shapelet ensemble for multi-dimensional time series, *in* 'SDM'.

Chakrabarti, K., Keogh, E., Mehrotra, S. and Pazzani, M. (2002), 'Locally adaptive dimensionality reduction for indexing large time series databases', *ACM Trans. Database Syst.* **27**(2), 188–228.

Chang, K.-W., Deka, B., Hwu, W.-M. W. and Roth, D. (2012), Efficient pattern-based time series classification on gpu, *in* 'Proceedings of the 12th IEEE International Conference on Data Mining'.

Ghalwash, M. and Obradovic, Z. (2012), 'Early classification of multivariate temporal observations by extraction of interpretable shapelets', *BMC Bioinformatics* **13**(1).
URL: *http://dx.doi.org/10.1186/1471-2105-13-195*

Grabocka, J., Schilling, N., Wistuba, M. and Schmidt-Thieme, L. (2014), Learning time-series shapelets, *in* 'Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '14, ACM, New York, NY, USA, pp. 392–401.
URL: *http://doi.acm.org/10.1145/2623330.2623613*

Guyon, I. and Elisseeff, A. (2003), 'An introduction to variable and feature selection', *J. Mach. Learn. Res.* **3**, 1157–1182.

Hartmann, B. and Link, N. (2010), Gesture recognition with inertial sensors and optimized dtw prototypes, *in* 'IEEE International Conference on Systems Man and Cybernetics'.

Hartmann, B., Schwab, I. and Link, N. (2010), Prototype optimization for temporarily and spatially distorted time series, *in* 'the AAAI Spring Symposia'.

He, Q., Zhuang, F., Shang, T., Shi, Z. et al. (2012), Fast time series classification based on infrequent shapelets, *in* '11th IEEE International Conference on Machine Learning and Applications'.

Hills, J., Lines, J., Baranauskas, E., Mapp, J. and Bagnall, A. (2013), 'Classification of time series by shapelet transformation', *Data Mining and Knowledge Discovery* .

[2] `www.reduction-project.eu`

Keogh, E., Zhu, Q., Hu, B., Y., H., Xi, X., Wei, L. and Ratanamahatana, C. A. (2011), 'The UCR Time Series Classification/Clustering Homepage', `www.cs.ucr.edu/~eamonn/time_series_data/`. [Online; accessed 02-March-2014].

Lines, J. and Bagnall, A. (2012), Alternative quality measures for time series shapelets, *in* 'Intelligent Data Engineering and Automated Learning', Vol. 7435 of *Lecture Notes in Computer Science*, pp. 475–483.

Mueen, A., Keogh, E. and Young, N. (2011), Logical-shapelets: an expressive primitive for time series classification, *in* 'Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'.

Rakthanmanon, T. and Keogh, E. (2013), 'Fast shapelets: A scalable algorithm for discovering time series shapelets', *Proceedings of the 13th SIAM International Conference on Data Mining* .

Sivakumar, P. and Shajina, T. (2012), Human gait recognition and classification using time series shapelets, *in* 'IEEE International Conference on Advances in Computing and Communications'.

Williams, B., Toussaint, M. and Storkey, A. (2006), Extracting motion primitives from natural handwriting data, *in* S. Kollias, A. Stafylopatis, W. Duch and E. Oja, eds, 'Artificial Neural Networks ICANN 2006', Vol. 4132 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 634–643.

Xing, Z., Pei, J. and Yu, P. (2012), 'Early classification on time series', *Knowledge and information systems* **31**(1), 105–127.

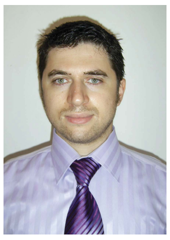
Xing, Z., Pei, J., Yu, P. and Wang, K. (2011), 'Extracting interpretable features for early classification on time series', *Proceedings of the 11th SIAM International Conference on Data Mining* .

Ye, L. and Keogh, E. (2009), Time series shapelets: a new primitive for data mining, *in* 'Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'.

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

Zakaria, J., Mueen, A. and Keogh, E. (2012), Clustering time series using unsupervised-shapelets, *in* 'Proceedings of the 12th IEEE International Conference on Data Mining'.

Author Biographies



Josif Grabocka is a doctoral candidate affiliated with the Information Systems and MachineLearning Lab belonging to the Institute of Computer Science at University of Hildesheim, Germany. He obtained his Diploma in Computer Engineering at the Middle East Technical University of Ankara, Turkey in 2007 and received a Master degree in Applied Artificial Intelligence from the University of Dalarna, Sweden in 2010. The primary research interests are Data Mining and Machine Learning and specifically mining time-series data. He has published articles in recognized journals on the field such as IEEE TKDE and Journal of Data Mining (DMKD/DAMI), as well as top conferences, such as ECML and ACM KDD.

Martin Wistuba is a doctoral candidate in the Information Systems and Machine Learning Lab, University of Hildesheim, Germany. He received his Master degree in Computer Science from the University of Paderborn, Germany in 2012. Wistuba has conducted research on applying factorization models to Artificial Intelligence domains. In addition, Martin Wistuba has worked on automatic hyper-parameter tuning for general machine learning algorithms. His research on hyper-parameters has been published in top conferences, such as IEEE ICDM or ECML.

Lars Schmidt-Thieme is a full professor leading the ISML Lab, University of Hildesheim, Germany. He obtained his Diploma in Mathematics at the University of Heidelberg in 1999 and received his PhD at the Department of Economics and Business Engineering at the University of Karlsruhe, Germany, in 2003. From 2003 to 2006 he was a professor at the Institute for Computer Science at the University of Freiburg, Germany. His main research interests are Machine Learning and Data Mining, especially classification and regression problems as well as pattern mining for complex data. He has published articles in top international conferences proceedings (e.g. IEEE ICDM, ACM KDD, UAI) and journals. He is member of program committees of international top conferences (e.g. ACM KDD, SIAM SDM, ECML) as well as executive board member of the German Classification Society.