A Time Series Forest for Classification and Feature Extraction[☆]

Houtao Deng[*]

Intuit, Mountain View, CA, USA

George Runger

Arizona State University, Tempe, AZ, USA

Eugene Tuv, Martyanov Vladimir

Intel, Chandler, AZ, USA

Abstract

A tree-ensemble method, referred to as time series forest (TSF), is proposed for time series classification. TSF employs a combination of entropy gain and a distance measure, referred to as the Entrance (entropy and distance) gain, for evaluating the splits. Experimental studies show that the Entrance gain improves the accuracy of TSF. TSF randomly samples features at each tree node and has computational complexity linear in the length of time series, and can be built using parallel computing techniques. The temporal importance curve is proposed to capture the temporal characteristics useful for classification. Experimental studies show that TSF using simple features such as mean, standard deviation and slope is computationally efficient and outperforms strong competitors such as one-nearest-neighbor classifiers with dynamic time warping.

Keywords: decision tree; ensemble; Entrance gain; interpretability; large margin; time series classification;

[*]Corresponding author: hdeng3@asu.edu

Email addresses: hdeng3@asu.edu (Houtao Deng), george.runger@asu.edu (George Runger), eugene.tuv@intel.com (Eugene Tuv), vladimir.martyanov@intel.com (Martyanov Vladimir)

1. Introduction

Time series classification has been playing an important role in many disciplines such as finance [25] and medicine [2]. Although one can treat the value of each time point as a feature and use a regular classifier such as one-nearest-neighbor (NN) with Euclidean distance for time series classification, the classifier may be sensitive to the distortion of the time axis and can lead to unsatisfactory accuracy performance. One-nearest-neighbor with dynamic time warping (NNDTW) is robust to the distortion of the time axis and has proven exceptionally difficult to beat [20]. However, NNDTW provides limited insights into the temporal characteristics useful for distinguishing time series from different classes.

The temporal features calculated over time series intervals [15], referred to as interval features, can capture the temporal characteristics, and can also handle the distortion in the time axis. For example, in the two-class time series shown in Figure 1, the time series from one of the classes have sudden changes between time 201 and time 400 but not in the same time points. An interval feature such as the standard deviation between time 201 and time 400 is able to distinguish the two-class time series.

Previous work [15] has built decision trees on interval features. However, a large number of interval features can be extracted from time series, and there can be a large number of candidate splits to evaluate at each tree node. Class-based measures (e.g., entropy gain), which evaluate the ability of separating the classes, are commonly used to select the best split in a node. However, there can be many splits having the same ability of separating the classes. Therefore, measures able to further distinguish these splits are desirable. Also, given a large number of features/splits, an efficient and accurate classifier that can provide insights into the temporal characteristics is valuable.

To this end, we propose a novel tree-ensemble classifier: time series forest (TSF). TSF employs a new measure called the *Entrance* (entropy and distance) gain to identify high-quality splits. We show that TSF using Entrance gain outperforms TSF using entropy gain and also two NNDTW algorithms. By using a random feature sampling strategy, TSF has computational complexity linear in the time series length. Furthermore, we propose the temporal importance curve to capture the temporal characteristics informative for time series classification.

The remainder of this paper is organized as follows. Section 2 presents the

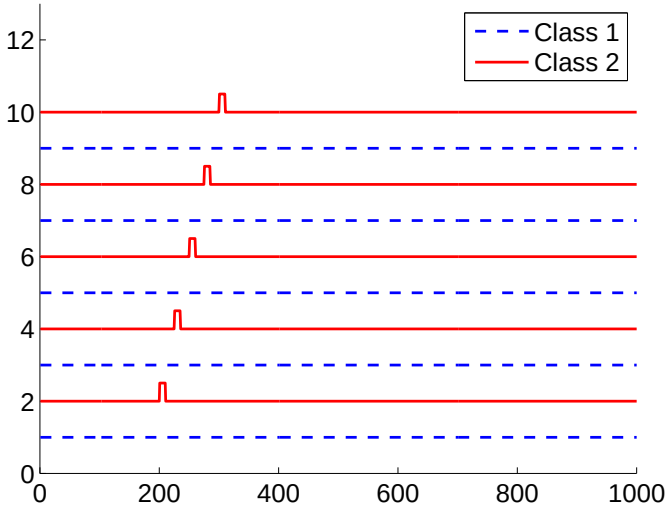


Figure 1: The time series from class 2 have sudden changes between time 201 and time 400. An interval feature such as the standard deviation between time 201 and time 400 can distinguish the time series from the two classes.

definition of the problem and related work. Section 3 introduces the interval features. Section 4 describes the TSF method. Section 5 demonstrates the effectiveness and efficiency of TSF by experiments. Conclusions are drawn in Section 6.

2. Definition and Related Work

Given N training time series instances (examples): $\{e_1, ..., e_i, ..., e_N\}$ and the corresponding class labels $\{y_1, ..., y_i, ..., y_N\}$, where $y_i \in \{1, 2, ..., C\}$, the objective of time series classification is to predict the class labels for testing instances. Here we assume the values of time series are measured at equally-spaced intervals, and also assume the training and testing time series instances are of the same length M.

Time series classification methods can be divided into instance-based and feature-based methods. Instance-based classifiers predict a testing instance based on its similarity to the training instances. Among instance-based classifiers, nearest-neighbor classifiers with Euclidean distance (NNEuclidean) or dynamic time warping (NNDTW) have been widely and successfully used

[12, 21, 8, 24]. Usually NNDTW performs better than NNEuclidean (dynamic time warping [17] is robust to the distortion in the time axis), and is considered as a strong solution for time series problems [13]. Instance-based classifiers can be accurate, but they provide limited insights into the temporal characteristics useful for classification.

Feature-based classifiers build models on temporal features, and potentially can be more interpretable than instance-based classifiers. Feature-based classifiers commonly consist of two steps: defining the temporal features and training a classifier based on the temporal features defined. Nanopoulos et al. [11] extracted statistical features such as the mean and deviation of an entire time series, and then used a multi-layer perceptron neural network for classification. This method only captured the global properties of time series. Local properties, potentially informative for classification, were ignored. Geurts [7] extracted local temporal properties after discretizing the time series. Rodríguez et al. [15] boosted binary stumps on temporal features from intervals of the time series and Rodríguez and Alonso [14], Rodríguez et al. [16] applied classifiers such as a decision tree and a SVM on the temporal features extracted from the boosted binary stumps. However, only binary stumps were boosted, and the effect of using more complex base learners, such as decision trees, should be studied [15] (but larger tree models impact the computational complexity). Furthermore, in decision trees [15, 14, 16] class-based measures are often used to evaluate the candidate splits in a node. However, the number of candidate splits is generally large, and, thus, there can be multiple splits having the same ability of separating the classes. Consequently, additional measures able to further distinguish these features are desirable. Ye and Keogh [23] briefly discussed strategies of introducing additional measures to break ties, but it was in a different context.

Recently, Ye and Keogh [23] proposed time series shapelets to perform interpretable time series classification. Shapelets are time series subsequences which are in some sense maximally representative of a class [23]. Ye and Keogh [23], Xing et al. [22], Lines et al. [10] have successfully shown that time series shapelets can produce highly interpretable results. In term of accuracy, Lines et al. [10] showed that the shapelet approach is comparable to NNDTW for nine data sets investigated.

Eruhimov et al. [5] considered a massive number of features. The feature sets were derived from statistical moments, wavelets, Chebyshev coefficients, PCA coefficients, and the original values of time series. The method can be accurate, but is hard to interpret and computationally expensive. The

objective of our work is to produce an effective and efficient classifier that uses/yields a set of simple features that can contribute to the domain knowledge. For example, in manufacturing applications, specific properties of the time series signals that discriminate conforming from un-conforming products are invaluable to diagnose, correct, and improve processes.

3. Interval Features

Interval features are calculated from a time series interval, e.g., "the interval between time 10 and time 30". Many types of features over a time interval can be considered, but one may prefer simple and interpretable features such as the mean and standard deviation, e.g., "the average of the time series segment between time 10 and time 30".

Let K be the number of feature types and $f_k(\cdot)$ $(k = 1, 2, ..., K)$ be the k^{th} type. Here we consider three types: $f_1 = mean$, $f_2 = standard\ deviation$, $f_3 = slope$. Let $f_k(t_1, t_2)$ for $1 \leq t_1 \leq t_2 \leq M$ denote the k^{th} interval feature calculated over the interval between t_1 and t_2. Let v_i be the value at time i for a time series example. Then the three interval features for the example are calculated as follows:

$$f_1(t_1, t_2) = \frac{\sum_{i=t_1}^{t_2} v_i}{t_2 - t_1 + 1} \tag{1}$$

$$f_2(t_1, t_2) = \begin{cases} \sqrt{\frac{\sum_{i=t_1}^{t_2} (v_i - f_1(t_1,t_2))^2}{t_2 - t_1}} & t_2 > t_1 \\ 0 & t_2 = t_1 \end{cases} \tag{2}$$

$$f_3(t_1, t_2) = \begin{cases} \hat{\beta} & t_2 > t_1 \\ 0 & t_2 = t_1 \end{cases} \tag{3}$$

where $\hat{\beta}$ is the slope of the least squares regression line of the training set $\{(t_1, v_{t_1}), (t_1 + 1, v_{t_1+1}) \ldots, (t_2, v_{t_2})\}$.

Interval features have been shown to be effective for time series classification [15, 14, 16]. However, the interval feature space is large ($O(M^2)$). Rodríguez et al. [15] considered using only intervals of lengths equal to powers of two, and, therefore, reduced the feature space to $O(M \log M)$. Here we consider the random sampling strategy used in a random forest [1] that reduces the feature space to $O(M)$ at each tree node.

4. Time Series Forest Classifier

4.1. Splitting criterion

A time series tree is the base component of a time series forest, and the splitting criterion is used to determine the best way to split a node in a tree. A candidate split S in a time series tree node tests the following condition (for simplicity and without loss of generality, we assume the root node here):

$$f_k(t_1, t_2) \leq \tau \tag{4}$$

for a threshold τ. The instances satisfying the condition are sent to the left child node. Otherwise, the instances are sent to the right child node.

Let $\{f_k^n(t_1, t_2), n \in 1, 2, ..., N\}$ denote the set of values of $f_k(t_1, t_2)$ for all training instances at the node. To obtain a good threshold τ in equation 4, one can sort the feature values of all the training instances and then select the best threshold from the midpoints between pairs of consecutive values, but this can be too costly [14]. We consider the strategy employed in Rodríguez and Alonso [14]. The candidate thresholds for a particular type feature f_k are formed such that the range of $[\min_{n=1}^{N}(f_k^n(t_1, t_2)),\ \max_{n=1}^{N}(f_k^n(t_1, t_2))]$ is divided into equal-width intervals. The number of candidate thresholds is denoted as κ and is fixed, e.g., 20. The best threshold is then selected from the candidate thresholds. In this manner, sorting is avoided, and only κ tests are needed.

Furthermore, a splitting criterion is needed to define the best split S^*: $f_*(t_1^*, t_2^*) \leq \tau^*$. We employ a combination of entropy gain and a distance measure as the splitting criterion. Entropy gain are commonly used as the splitting criterion in tree models. Denote the proportions of instances corresponding to classes $\{1, 2, ..., C\}$ at a tree node as $\{\gamma_1, \gamma_2, ..., \gamma_C\}$, respectively. The entropy at the node is defined as

$$Entropy = -\Sigma_{c=1}^{C} \gamma_c \log \gamma_c \tag{5}$$

The entropy gain $\triangle Entropy$ for a split is then the difference between the weighted sum of entropy at the child nodes and the entropy at the parent node, where the weight at a child node is the proportion of instances assigned to that child node.

$\triangle Entropy$ evaluates the usefulness of separating the classes. However, in time series classification, the number of candidate splits can be large, and there are often cases where multiple candidate splits have the same

$\triangle Entropy$. Therefore we consider an additional measure called $Margin$, which calculates the distance between a candidate threshold and its nearest feature value. The $Margin$ of split $f_k(t_1, t_2) \leq \tau$ is calculated as

$$Margin = \min_{n=1,2,...,N} |f_k^n(t_1, t_2) - \tau| \tag{6}$$

where $f_k^n(t_1, t_2)$ is the value of $f_k(t_1, t_2)$ for the n^{th} instance at the node. A new splitting criterion E, referred to as the *Entrance* (entropy and distance) gain, is defined as a combination of $\triangle Entropy$ and $Margin$.

$$E = \triangle Entropy + \alpha \cdot Margin \tag{7}$$

where α is small enough so that the only role for α in the model is to break ties that can occur from the entropy gain alone. Alternatively, one can store the values of $\triangle Entropy$ and $Margin$ for a split, and use $Margin$ to break ties when another split has the same $\triangle Entropy$.

Clearly, the split with the maximum E should be selected to split the node. Furthermore, $Margin$ and E are sensitive to the scale of the features, and we employ the following strategy if different types of features have different scales. For each feature type f_k, select the split with the maximum Entrance gain. To compare the best splits from different feature types, the split with the maximum $\triangle Entropy$ is selected. If the best splits from different feature types have the same maximum $\triangle Entropy$, one of the best splits is randomly selected.

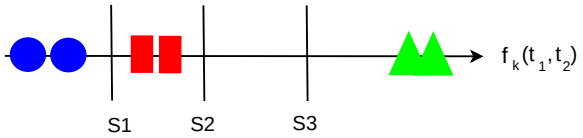


Figure 2: Here the x-axis represents the value of an interval feature. The figure shows six instances associated with three classes (blue, red, and green), and three splits (S_1, S_2, and S_3) producing the same entropy gain. The Entrance gain E is able to select S_3 as the best split.

Figure 2 illustrates the intuition behind the criterion E. The figure shows, in one dimension, six instances from three classes in different symbols/colors. Three candidate splits S_1, S_2 and S_3 are also shown in the figure. Clearly, all splits have the same $\triangle Entropy$, but one may prefer S_3 because S_3 has a larger margin than S_1 and S_2. The Entrance gain is able to choose S_3 as the best split.

Algorithm 1 *sample*() function: randomly samples a set of intervals $< T_1, T_2 >$, where T_1 is the set of starting time points of intervals, and T_2 is the set of ending points. The function $RandSampNoRep(set, samplesize)$ randomly selects *samplesize* elements from *set* without replacement.

$T_1 = \emptyset$, $T_2 = \emptyset$
$W = RandSampNoRep(\{1, ..., M\}, \sqrt{M})$
for w in set W **do**
  $T_1 = RandSampNoRep(\{1, ..., M - w + 1\}, \sqrt{M - w + 1})$
  for t_1 in set T_1 **do**
    $T_2 = T_2 \bigcup (t_1 + w - 1)$
  end for
end for
return $< T_1, T_2 >$

Algorithm 2 $tree(data)$: Time series tree. For simplicity of the algorithm, we assume different types of features are on the same scale so that E can be compared.

$< T_1, T_2 >$=*sample*()
calculate $Threshold_k$, the set of candidate thresholds for each feature type k
$E^* = 0$, $\triangle Entropy^* = 0$, $t_1^* = 0$, $t_2{}^* = 0$, $\tau^* = 0$, $f_* = \emptyset$
for $< t_1, t_2 >$ in set $< T_1, T_2 >$ **do**
  for k in 1:K **do**
    for τ in $Threshold_k$ **do**
      calculate $\triangle Entropy$ and E for $f_k(t_1, t_2) \leq \tau$
      if $E > E^*$ **then**
        $E^* = E$, $\triangle Entropy^* = \triangle Entropy$, $t_1^* = t_1$, $t_2^* = t_2$, $\tau^* = \tau$, $f_* = f_k$
      end if
    end for
  end for
end for
if $\triangle Entropy^* = 0$ **then**
  label this node as a leaf and return
end if
$data_{left} \leftarrow$ time series with $f_*(t_1^*, t_2^*) \leq \tau^*$
$data_{right} \leftarrow$ time series with $f_*(t_1^*, t_2^*) > \tau^*$
$tree(data_{left})$
$tree(data_{right})$

4.2. Time Series Tree and Time Series Forest

The construction of a time series tree follows a top-down, recursive strategy similar to standard decision tree algorithms, but uses the Entrance gain as the splitting criterion. Furthermore, the random sampling strategy employed in random forest (RF) [1] is considered here. At each node, RF only tests $\sqrt{p}$ features randomly sampled from the complete feature set consisting of p features. In each time series tree node, we consider randomly sampling $O(\sqrt{M})$ interval sizes and $O(\sqrt{M})$ starting positions. Therefore, the feature space is reduced to only $O(M)$. The sampling algorithm is illustrated in Algorithm 1.

The time series tree algorithm is shown in Algorithm 2. For simplicity, we assume different types of features are on the same scale so that E can be compared. If different types of features have different scales, the previous mentioned strategy can be used, that is, for each feature type f_k, select the split with the maximum Entrance gain. To compare the best splits from different feature types, the split with the maximum $\triangle Entropy$ is selected. Furthermore, a node is labeled as a leaf if there is no improvement on the entropy gain (e.g. all features have the same value or all instances belong to the same class).

A time series forest (TSF) is a collection of time series trees. A TSF predicts a testing instance to be the majority class according to the votes from all time series trees.

4.3. Computational Complexity

Let n_j^i denote the number of instances in the j^{th} node at the i^{th} depth in a time series tree. At each node, calculating the splitting criterion of a single interval feature has complexity $O(n_j^i \kappa)$, where κ is the number of candidate thresholds. As $O(M)$ interval features are randomly selected for evaluation, the complexity for evaluating the features at a node is $O(n_j^i M \kappa)$. As κ is considered as a constant, the complexity at a node is $O(n_j^i M)$.

The total number of instances at each depth is at most N (i.e., $\sum_j n_j^i \leq N$). Therefore, at the i^{th} depth in the tree, the complexity is $O(\sum_j n_j^i M) \leq O(NM)$. Assuming the maximum depth of a tree model is $O(\log N)$ [19], the complexity of a time series tree becomes $O(MN \log N)$. Therefore, the complexity of a TSF with $nTree$ time series trees is at most $O(nTreeMN \log N)$, linear in the length of time series.

4.4. Temporal Importance Curve

TSF consists of multiple trees and is difficult to understand. Here we propose the temporal importance curve to provide insights into time series classification. At each node of TSF, the entropy gain can be calculated for the interval feature used for splitting. For a time index in the time series, one can add the entropy gain of all the splits associated with the time index for a particular type of feature. That is, for a feature type f_k, the importance score for time index t can be calculated as

$$Imp_k(t) = \sum_{t_1 \le t \le t_2, \nu \in SN} \triangle Entropy(f_k(t_1, t_2), \nu) \tag{8}$$

where SN is the set of split nodes in TSF, and $\triangle Entropy(f_k(t_1, t_2), \nu)$ is the entropy gain for feature $f_k(t_1, t_2)$ at node ν. Note $\triangle Entropy(f_k(t_1, t_2), \nu) = 0$ if $f_k(t_1, t_2)$ is not used for splitting node ν. Furthermore, one temporal importance curve is generated for each feature type. Consequently, for the mean, standard deviation and slope features, we calculate the mean, standard deviation, and slope temporal importance curves, respectively.

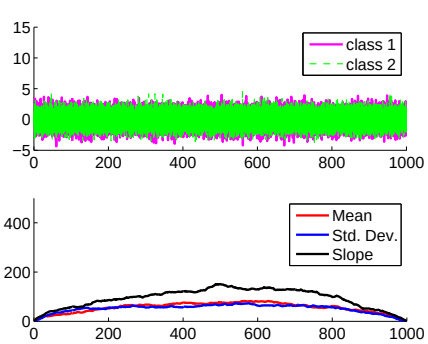
To investigate the temporal importance curve, we simulated two data sets, each with 1000 time points and two classes. For the first data set the time series have the same distribution so that no feature is useful for separating the classes. The time series values from both classes are normally distributed with zero mean and unit variance. The time series and the importance curves from TSF using Entrance gain are shown in Figure 3(a). It can be seen that all curves have larger values in the middle.

Note that the number of intervals that include time index t in a time series is

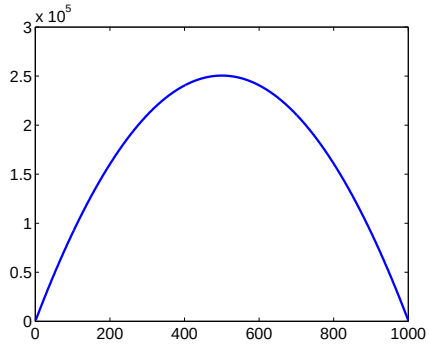
$$Num(t) = t(M - t + 1) \tag{9}$$

Consequently, different time indices are associated with different numbers of intervals. The number of intervals for each time index for time series with 1000 time points is plotted in Figure 3(b). The indices in the middle have more intervals than the indices on the edges of the time series. Because $Imp_k(t)$ is calculated by adding the entropy gain of all the splits associated with time index t for feature f_k, it can be biased towards the time points having more interval features (particularly if no feature is important for classification).

For the second data set the time series from the two classes have different means in interval [201, 250], and different standard deviations in interval

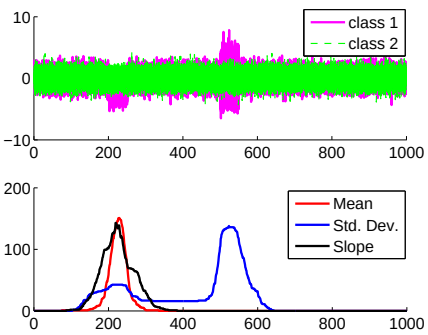


(a) The time series data and the importance curves from TSF.

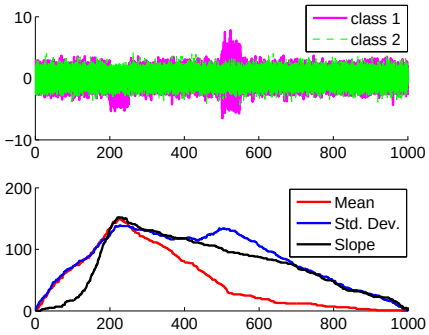


(b) The number of intervals associated with each time index. The time indices in the middle are contained in more intervals.

Figure 3: When no feature is important for classification, the curves may be expected to have larger values for the middle indices as there are more intervals associated with the middle indices.



(a) The time series and the temporal importance curves obtained from TSF using Entrance gain.



(b) The time series and the temporal importance curves obtained from TSF using entropy gain.

Figure 4: The time series from the two classes differ in the mean in interval [201, 250], and differ in the standard deviation in interval [501, 550]. The importance curves from TSF using Entrance gain are able to capture the informative intervals well. The curves from TSF using entropy gain have peaks in interval [201, 250], but have long tails.

[501, 550]. The temporal importance curves from TSF using Entrance gain are shown in Figure 4(a). The curves for the mean and slope have peaks in interval [201, 250], and the curve for the standard deviation has a peak in interval [501, 550]. Therefore, these curves capture the important temporal characteristics.

We also built TSF using entropy gain, and the corresponding temporal importance curves are shown in Figure 4(b). Although the curves also have peaks in interval [201, 250], the curves have long tails. Indeed, the entropy gain is not able to distinguish many interval features. For example, the mean feature for interval [201,250], and the mean feature for interval [201,400] have the same entropy gain as both can distinguish the two classes of time series. However, the mean feature for interval [201,250] has a larger E than the mean feature for interval [201,400]. Consequently, TSF using Entrance gain is able to capture the temporal characteristics more accurately.

5. Experiments

5.1. Experimental Setup

The main functions of the TSF algorithm were implemented in Matlab, while computationally expensive subfunctions such as interval feature calculations were written in C. The parameters were set as follows: the number of trees = 500, $f(\cdot) = \{mean, standard\ deviation, slope\}$, and the number of candidate thresholds $\kappa = 20$. TSF was applied to a set of time series benchmark data sets [9] summarized in Table 1. The training/testing split setting is the same as in Keogh et al. [9]. The experiments were run on a computer with four cores and the TSF algorithm was built in parallel.

The purpose of the experiments is to answer the following questions: (1) Does the Entrance gain criterion improve the accuracy performance and how is the accuracy performance of TSF compared to other time series classifiers? (2) Is TSF computationally efficient? (3) Can the temporal importance curves provide some insights about the temporal characteristics useful for classification?

5.2. Results

We investigated the performance of TSF using the Entrance gain criterion (denoted as TSF) and using the original entropy gain criterion (denoted as TSF-entropy), respectively. We also considered alternative classifiers for comparison: random forest [1] applied to the interval features with sizes

	Length	Training instances	Testing instances	Classes
50words	270	450	455	50
Adiac	176	390	391	37
Beef	470	30	30	5
CBF	128	30	900	3
ChlorineConcentration	166	467	3840	3
CinC_ECG_torso	1639	40	1380	4
Coffee	286	28	28	2
Cricket_X	300	390	390	12
Cricket_Y	300	390	390	12
Cricket_Z	300	390	390	12
DiatomSizeReduction	345	16	306	4
ECG200	96	100	100	2
ECGFiveDays	136	23	861	2
FaceAll	131	560	1690	14
FaceFour	350	24	88	4
FacesUCR	131	200	2050	14
Fish	463	175	175	7
GunPoint	150	50	150	2
Haptics	1092	155	308	5
InlineSkate	1882	100	550	7
ItalyPowerDemand	24	67	1029	2
Lighting2	637	60	61	2
Lighting7	319	70	73	7
MALLAT	1024	55	2345	8
MedicalImages	99	381	760	10
MoteStrain	84	20	1252	2
NonInvasiveFatalECG_Thorax1	750	1800	1965	42
NonInvasiveFatalECG_Thorax2	750	1800	1965	42
OliveOil	570	30	30	4
OSULeaf	427	200	242	6
SonyAIBORobotSurface	70	20	601	2
SonyAIBORobotSurfaceII	65	27	953	2
StarLightCurves	1024	1000	8236	3
SwedishLeaf	128	500	625	15
Symbols	398	25	995	6
Syntheticcontrol	60	300	300	6
Trace	275	100	100	4
TwoLeadECG	82	23	1139	2
TwoPatterns	128	1000	4000	4
uWaveGestureLibrary_X	315	896	3582	8
uWaveGestureLibrary_Y	315	896	3582	8
uWaveGestureLibrary_Z	315	896	3582	8
Wafer	152	1000	6164	2
WordsSynonyms	270	267	638	25
Yoga	426	300	3000	2

Table 1: Summary of the time series data sets: the number of training and testing instances, the number of classes and the lengths of the time series.

power of two (interRF), the 1-nearest-neighbor (NN) classifier with Euclidean distance (NNEuclidean), the 1-NN Best warping window DTW (DTWBest)

	TSF Entrance	**TSF entropy**	**interRF**	**NN Euclidean**	**DTW Best**	**DTW NoWin**
50words	0.2659	0.2769	0.2989	0.3690	0.2420	0.3100
Adiac	0.2302	0.2609	0.2506	0.3890	0.3910	0.3960
Beef	0.2333	0.3000	0.3000	0.4670	0.4670	0.5000
CBF	0.0256	0.0389	0.0411	0.1480	0.0040	0.0030
ChlorineConcentration	0.2537	0.2596	0.2273	0.3500	0.3500	0.3520
CinC_ECG_torso	0.0391	0.0688	0.1065	0.1030	0.0700	0.3490
Coffee	0.0357	0.0714	0.0000	0.2500	0.1790	0.1790
Cricket_X	0.2897	0.2872	0.3128	0.4260	0.2360	0.2230
Cricket_Y	0.2000	0.2000	0.2436	0.3560	0.1970	0.2080
Cricket_Z	0.2436	0.2385	0.2436	0.3800	0.1800	0.2080
DiatomSizeReduction	0.0490	0.1013	0.0980	0.0650	0.0650	0.0330
ECG200	0.0800	0.0700	0.1700	0.1200	0.1200	0.2300
ECGFiveDays	0.0557	0.0697	0.1231	0.2030	0.2030	0.2320
FaceAll	0.2325	0.2314	0.2497	0.2860	0.1920	0.1920
FaceFour	0.0227	0.0341	0.0568	0.2160	0.1140	0.1700
FacesUCR	0.1010	0.1088	0.1283	0.2310	0.0880	0.0951
Fish	0.1543	0.1543	0.1486	0.2170	0.1600	0.1670
GunPoint	0.0467	0.0467	0.0400	0.0870	0.0870	0.0930
Haptics	0.5520	0.5649	0.5487	0.6300	0.5880	0.6230
InlineSkate	0.6818	0.6746	0.6873	0.6580	0.6130	0.6160
ItalyPowerDemand	0.0301	0.0330	0.0321	0.0450	0.0450	0.0500
Lighting2	0.1803	0.1803	0.2459	0.2460	0.1310	0.1310
Lighting7	0.2603	0.2603	0.2740	0.4250	0.2880	0.2740
MALLAT	0.0448	0.0716	0.0644	0.0860	0.0860	0.0660
MedicalImages	0.2237	0.2316	0.2658	0.3160	0.2530	0.2630
MoteStrain	0.1190	0.1182	0.0942	0.1210	0.1340	0.1650
NonInvasiveFatalECG_Thorax1	0.0987	0.1033	0.1104	0.1710	0.1850	0.2090
NonInvasiveFatalECG_Thorax2	0.0865	0.0936	0.0875	0.1200	0.1290	0.1350
OliveOil	0.0667	0.1000	0.1333	0.1330	0.1670	0.1330
OSULeaf	0.4339	0.4256	0.4587	0.4830	0.3840	0.4090
SonyAIBORobotSurface	0.2330	0.2346	0.2562	0.1410	0.1410	0.1690
SonyAIBORobotSurfaceII	0.1868	0.1773	0.2067	0.3050	0.3050	0.2750
StarLightCurves	0.0357	0.0364	0.0327	0.1510	0.0950	0.0930
SwedishLeaf	0.1056	0.1088	0.0768	0.2130	0.1570	0.2100
Symbols	0.1116	0.1206	0.1216	0.1000	0.0620	0.0500
Syntheticcontrol	0.0267	0.0233	0.0167	0.1200	0.0170	0.0070
Trace	0.0200	0.0000	0.0400	0.2400	0.0100	0.0000
TwoLeadECG	0.1177	0.1115	0.1773	0.2530	0.1320	0.0960
TwoPatterns	0.0543	0.0530	0.0153	0.0900	0.0015	0.0000
uWaveGestureLibrary_X	0.2102	0.2127	0.2094	0.2610	0.2270	0.2730
uWaveGestureLibrary_Y	0.2876	0.2881	0.3023	0.3380	0.3010	0.3660
uWaveGestureLibrary_Z	0.2624	0.2669	0.2764	0.3500	0.3220	0.3420
Wafer	0.0054	0.0047	0.0071	0.0050	0.0050	0.0200
WordsSynonyms	0.3793	0.3809	0.4138	0.3820	0.2520	0.3510
Yoga	0.1513	0.1567	0.1380	0.1700	0.1550	0.1640
win/lose/tie	**-**	**16/28/1**	**13/32/0**	**4/41/0**	**17/28/0**	**16/29/0**
Average rank	**2.48**	**2.86**	**3.43**	**5.04**	**3.31**	**3.88**
Rank difference	**-**	**0.38**	**0.96**	**2.57**	**0.83**	**1.40**
Wilcoxon	**-**	**0.007**	**0.000**	**0.000**	**0.065**	**0.006**

Table 2: The error rates of TSF using the splitting criterion: Entrance gain (TSF) or entropy gain (TSF-entropy), random forest with 500 trees applied to the interval features with sizes power of two (interRF), 1-NN with Euclidean distance (NNEuclidean), 1-NN with the best warping window DTW (DTWBest) [12], and 1-NN DTW with no warping window (DTWNoWin). The win-lose-tie results of each competitor compared to TSF, the average rank of each classifier, the rank difference and the Wilcoxon signed ranks test between TSF and each competitor are also calculated. When multiple methods have the same error rate for a data set, the average rank is used. For example, both DTWBest and DTWNoWin have the minimum error rate 0.192 for the FaceAll data set, and, thus, the rank for both is 1.5.

[12] and the 1-NN DTW with no warping window (DTWNoWin) methods

acquired directly from Keogh et al. [9]. DTWBest has a fixed window limiting the window width and searches for the best window size, while DTWNoWin does not use such a window.

The classification error rates are shown in Table 2. To compare multiple classifiers to TSF over multiple data sets, we used the procedure for comparing multiple classifiers with a control over multiple data sets suggested by Demšar [3], i.e., the Friedman test [6] followed by the Bonferroni-Dunn test [4] if the Friedman test shows a significant difference between the classifiers. In our case, the Friedman test shows that there is a significant difference between the six classifiers at the 0.001 level. Therefore, we proceeded with the Bonferroni-Dunn test.

For the Bonferroni-Dunn test, the performance of two classifiers is different at the α level if the their average ranks differ by at least the critical difference (CD):

$$z_\alpha = q_\alpha \sqrt{\frac{N_{classifier}(N_{classifier}+1)}{6N_{data}}} \tag{10}$$

where $N_{classifier}$ is the number of classifiers in the comparison (six classifiers in our experiments), N_{data} is the number of data sets (45 data sets in our experiments), and q_α is the critical value for the two-tailed Bonferroni-Dunn test for multiple classifier comparison with a control. Note $q_{0.05} = 2.576$ and $q_{0.1} = 2.326$ (Table 5(b) in Demšar [3]), then according to Equation 10, $z_{0.05} = 1.016$ and $z_{0.1} = 0.917$. The average rank of each classifier, and the difference between the average ranks of TSF and each competitor are shown in Table 2. According to the rank difference, there is a significant difference between TSF and competitors NNEuclidean, DTWNoWin and interRF at the 0.1 level.

In addition to the multi-classifier comparison procedure, we also considered Wilcoxon signed ranks test [18] suggested for comparing a pair of classifiers, as the resolution for the multi-classifier comparison procedure can be too low to distinguish two classifiers with significantly different performance, but with close average ranks. For example, for six classifiers and 45 data sets, assume classifier A always ranks the first and classifier B always ranks the second. Although classifier A is always better than classifier B, the average ranks of classifier A and classifier B differ by only one, and therefore there is no significant difference between the two classifiers at the 0.05 level according to the two-tailed Bonferroni-Dunn test.

The p-values of the Wilcoxon signed ranks tests between TSF and each

competitor are shown in Table 2. It can be seen there is a significant difference between TSF and all other competitors: TSF-entropy, interRF, NNEuclidean, DTWNoWin and DTWBest at the 0.1 level.

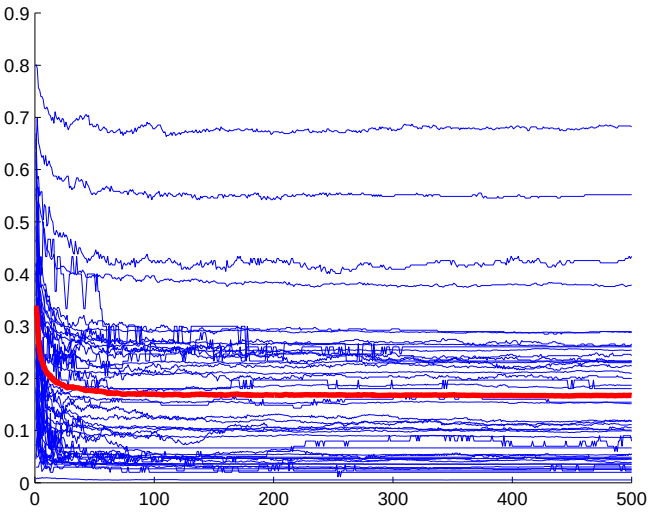
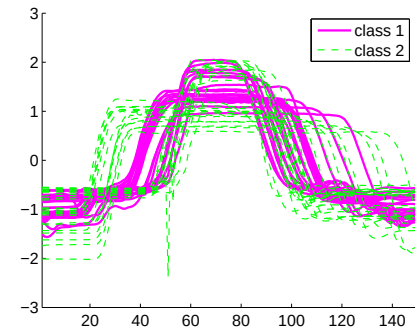


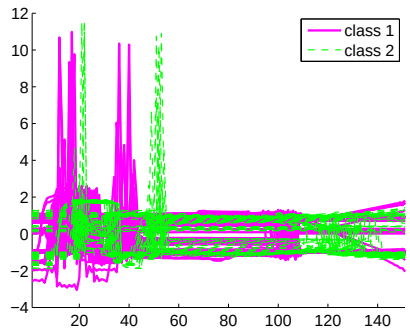
Figure 5: Plot of the error rate of each data set versus the number of trees in TSF, and the average error rate over all data sets versus the number of trees (represented by the thicker red line). We want to show the trend so different data sets are not distinguished. The error rates tend to decrease as the number of trees increases, but the change is relatively small for most data sets after 100 trees.

Next consider the robustness of TSF accuracy to the number of trees. Figure 5 shows the error rate of each data set versus the number of trees, and the average error rate over all data sets versus the number of trees (represented by the thicker red line). The error rates tend to decrease as the number of trees increases, but the change is relatively small for most data sets after 100 trees.

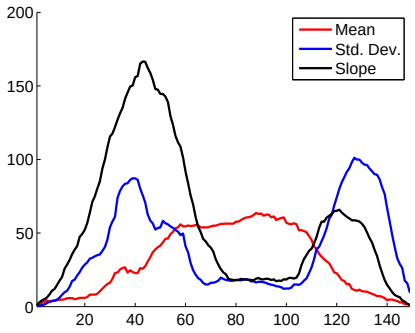
The GunPoint and Wafter time series and their corresponding temporal importance curves (mean, standard deviation and slope) are shown in Figure 6. For the GunPoint time series, the mean temporal importance curve captures the characteristic that the two classes have different means in interval [60,100]. The standard deviation and slope temporal importance curves, respectively, capture the characteristics that the two classes have different standard deviations and slopes in the left and right sides of the time se-



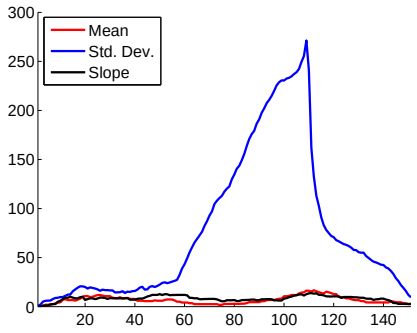
(a) GunPoint time series



(b) Wafer time series



(c) The temporal importance curves for the GunPoint data.



(d) The temporal importance curves for the Wafer data.

Figure 6: The time series and the temporal importance curves (mean, standard deviation and slope) for the GunPoint data set and the Wafer data set, respectively.

ries. For the Wafer time series, the standard deviation temporal importance curve captures the sudden changes of the time series of class 1 near the 100^{th} point. Consequently, the temporal importance curve is able to provide insights into the temporal characteristics useful for distinguishing time series from different classes.

5.3. Computational Complexity

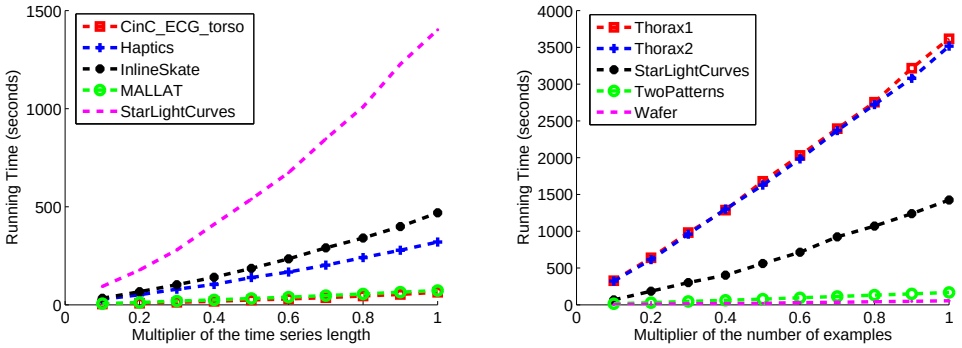
First consider the computational complexity of TSF with regard to the length of time series. We selected the data sets with more than 1000 time points. For each data set, λM of the time points were randomly sampled, where M is the length of the time series, and λ is a multiplier. The computational times for different values of λ are shown in Figure 7(a). Next consider the computational complexity of TSF with regard to the number of training instances. Data sets with more than 1000 training instances were selected. For each data set, λN of the time points were randomly sampled, where N is the number of training instances. The computational times for different values of λ are shown in Figure 7(b). It can be seen that the computational time tends to be linear both in the time series length and in the number of training instances.

Therefore, TSF is a computationally efficient classifier for time series. Furthermore, in the current TSF implementation, the interval features are dynamically calculated at each node, as pre-computing the interval features would need $O(M^2)$ features to be stored. It should noted, however, dynamic calculation can lead to repeated calculations of the interval features. Therefore, the implementation can be further improved by storing the interval features already calculated to avoid repeated calculations.

6. Conclusions

Both high accuracy and interpretability are desirable for classifiers. Previous classifiers such as NNDTW can be accurate, but provide limited insights into the temporal characteristics. Interval features can be used to capture temporal characteristics, however, the huge feature space can result in many splits having the same entropy gain. Furthermore, the computational complexity becomes a concern when the feature space becomes large.

Time series forest (TSF) proposed here addresses the challenges by using the following two strategies. Firstly, TSF uses a new splitting criterion named Entrance gain that combines the entropy gain and a distance measure to



(a) The computational time of TSF with regard to the time series length

(b) The computational time of TSF with regard to the number of training instances

Figure 7: The computational time of TSF with regard to the time series length and the number of training instances, respectively. Data sets with more than 1000 time points and 1000 training instances were selected, respectively. The computational time tends to be linear both in the time series length and in the number of training instances.

identify high-quality splits. Experimental studies on 45 benchmark data sets show that the Entrance gain improves the accuracy of TSF. Secondly, TSF randomly samples $O(M)$ features from $O(M^2)$ features, and thus makes the computational complexity linear in the time series length. In addition, each tree in TSF is grown independently, and, therefore, modern parallel computing techniques can be leveraged to speed up TSF.

TSF is an ensemble of trees and is not easy to understand. However, we propose the temporal importance curve, calculated from TSF, to capture the informative interval features. The temporal importance curve enables one to identify the important temporal characteristics.

TSF uses simple summary statistical features, but outperforms widely used alternatives. More complex features, such as wavelets, can be also used in the framework of TSF, which potentially can further improve the accuracy performance, but at the cost of interpretability.

In summary, TSF is an accurate, efficient time series classifier, and is able to provide insights on the temporal characteristics useful for distinguishing time series from different classes. We also note that TSF assumes that the time series are of the same length. Given a set of time series with different lengths, techniques such as dynamic time warping can be used to align the time series into the same length. Still, directly handling time series with varying lengths would make TSF more convenient to use, and future work

includes such an extension.

Acknowledgements

This research was partially supported by ONR grant N00014-09-1-0656. We also wish to thank the editor and anonymous reviewers for their valuable comments.

References

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

[2] I. Costa, A. Schönhuth, C. Hafemeister, A. Schliep, Constrained mixture estimation for analysis and robust classification of clinical time series, Bioinformatics 25 (2009) i6–i14.

[3] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine Learning Research 7 (2006) 1–30.

[4] O. Dunn, Multiple comparisons among means, Journal of the American Statistical Association (1961) 52–64.

[5] V. Eruhimov, V. Martyanov, E. Tuv, Constructing high dimensional feature space for time series classification, in: Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Springer, 2007, pp. 414–421.

[6] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1940) 86–92.

[7] P. Geurts, Pattern extraction for time series classification, in: Proceedings of the 5th European conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Springer, 2001, pp. 115–127.

[8] Y. Jeong, M. Jeong, O. Omitaomu, Weighted dynamic time warping for time series classification, Pattern Recognition 44 (2011) 2231–2240.

[9] E. Keogh, X. Xi, L. Wei, C. Ratanamahatana, The ucr time series classification/clustering. homepage: www.cs.ucr.edu/~eamonn/time_series_data/, 2006.

[10] J. Lines, L.M. Davis, J. Hills, A. Bagnall, A shapelet transform for time series classification, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD), ACM, 2012, pp. 289–297.

[11] A. Nanopoulos, R. Alcock, Y. Manolopoulos, Feature-based classification of time-series data, International Journal of Computer Research 10 (2001) 49–61.

[12] C. Ratanamahatana, E. Keogh, Making time-series classification more accurate using learned constraints, in: Proceedings of SIAM International Conference on Data Mining (SDM), SIAM, 2004, pp. 11–22.

[13] C. Ratanamahatana, E. Keogh, Three myths about dynamic time warping data mining, in: Proceedings of SIAM International Conference on Data Mining (SDM), SIAM, 2005, pp. 506–510.

[14] J. Rodríguez, C. Alonso, Interval and dynamic time warping-based decision trees, in: Proceedings of the 2004 ACM symposium on Applied computing, ACM, 2004, pp. 548–552.

[15] J. Rodríguez, C. Alonso, H. Boström, Boosting interval based literals, Intelligent Data Analysis 5 (2001) 245–262.

[16] J. Rodríguez, C. Alonso, J. Maestro, Support vector machines of interval-based features for time series classification, Knowledge-Based Systems 18 (2005) 171–178.

[17] H. Sakoe, Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing 26 (1978) 43–49.

[18] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (1945) 80–83.

[19] I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2005.

[20] X. Xi, E. Keogh, C. Shelton, L. Wei, C.A. Ratanamahatana, Fast time series classification using numerosity reduction, in: Proceedings of the 23rd international conference on Machine learning (ICML), ACM, 2006, pp. 1033–1040.

[21] Z. Xing, J. Pei, P. Yu, Early prediction on time series: a nearest neighbor approach, in: Proceedings of the 21st International Joint Conference on Artifical Intelligence (IJCAI), Morgan Kaufmann, 2009, pp. 1297–1302.

[22] Z. Xing, J. Pei, P.S. Yu, K. Wang, Extracting interpretable features for early classification on time series, in: Proceedings of SIAM International Conference on Data Mining (SDM), SIAM, 2011, pp. 247–258.

[23] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD), ACM, 2009, pp. 947–956.

[24] D. Yu, X. Yu, Q. Hu, J. Liu, A. Wu, Dynamic time warping constraint learning for large margin nearest neighbor classification, Information Sciences 181 (2011) 2787–2796.

[25] Z. Zeng, H. Yan, Supervised classification of share price trends, Information Sciences 178 (2008) 3943–3956.