ShiftTree: An Interpretable Model-Based Approach for Time Series Classification

Balázs Hidasi and Csaba Gáspár-Papanek

Budapest University of Technology and Economics,
Department of Telecommunication and Media Informatics,
{hidasi,gaspar}@tmit.bme.hu

Abstract. Efficient algorithms of time series data mining have the common denominator of utilizing the special time structure of the attributes of time series. To accommodate the information of time dimension into the process, we propose a novel instance-level cursor based indexing technique, which is combined with a decision tree algorithm. This is beneficial for several reasons: (a) it is insensitive to the time level noise (for example rendering, time shifting), (b) its working method can be interpreted, making the explanation of the classification process more understandable, and (c) it can manage time series of different length. The implemented algorithm named ShiftTree is compared to the well-known instance-based time series classifier 1-NN using different distance metrics, used over all 20 datasets of a public benchmark time series database and two more public time series datasets. On these benchmark datasets, our experiments show that the new model-based algorithm has an average accuracy slightly better than the most efficient instance-based methods, and there are multiple datasets where our model-based classifier exceeds the accuracy of instance-based methods. We also evaluated our algorithm via blind testing on the 20 datasets of the SIGKDD 2007 Time Series Classification Challenge. To improve the model accuracy and to avoid model overfitting, we provide forest methods as well.

Keywords: model-based time series classification, decision trees, forest building methods.

1 Introduction

With the spread of automatic data collection systems, the role of time series has been increasing in business intelligence applications in the domains of entertainment, industry and of mobile devices. Even though the traditional source of time series databases is the financial sector, due to the decrease in the pricing of sensors, more and more time series data are collected from everyday electrical devices.

For example, most new cellular phones and laptops have a gyroscope for the collection of acceleration data. By processing these time series data, hand gesture controlled interfaces can be built into many applications. There seems to have been an increase in the number of time series based applications on the end-user level. Time series data can also be found in the fields of medicine and

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part II, LNAI 6912, pp. 48–64, 2011.

biology (e.g.: the ECG(electrocardiographic signal)), finance, system monitoring and logistics.

Naturally, supervised and unsupervised learning tasks also appear connected to time series data. These data mining tasks can be organized into the following categories:

1. Data mining of single time series
   (a) Next value prediction in time series (e.g. Stock market prediction [3])
   (b) Clustering of segments of the time series (e.g. Time series subsequence clustering)
   (c) Classification of segments of the time series (e.g. Hand gesture recognition in accelerator data [15])
   (d) Motif (similar subsequences) discovery in a longer time series [14]
2. Data mining of multiple time series
   (a) Clustering of time series (e.g. Segmentation of customers of an electricity provider by clustering the time series of their charging)
   (b) Classification of time series (e.g. Analyzing heart function by classification of ECG signals [2])

The complexity of this hierarchy can be reduced if we consider the fact that Points 1.b and 1.c can be incorporated into Case 2. by the segmentation of the original time series.

The reason for the difficulty of these tasks is rooted in the multi-dimensional problem space and the special connection between the attributes (element or values of time series): the sequence of attributes (elements) carries information about the source entity. In the case of traditional vector-based data representation, there is no information in the order of attributes, but time series elements, which are close to each other, have special connection through the dimension of time. For example, if the values are shifted in a time series by one position (for example, value of attributes i is replaced by attributes i-1), then the classification label or cluster ID of the time series will probably stay the same. The effective algorithms of time series data mining typically have some additional aspect to handle the effect of this time-dimension structure. Our new method is capable of considering time level aspects of time series.

Our approach is a novel model-based classification method labeling different time series by learning from the database with unknown labels. We named this algorithm ShiftTree. The beneficial properties of the method are the following:

- accuracy level similar to other techniques
- interpretable model
- capable of handling datasets of time series with different lengths
- preprocessing not necessary
- expert knowledge can be built into the modeling process.
- correspondences coded in time dimension can be interpreted

The rest of this paper is organized as follows: Section 2 reviews the time series classification techniques, Section 4 presents the concept of our novel approach

ShiftTree, whereas its formal definition is described in Section 5. After the Section about interpretability, the Section 6 provides two other techniques to improve the accuracy. Section 7 summarizes the numerical results, finally, Section 8 sums up our experiments.

2 Related Works

Time series specific classification algorithms usually belong to two categories: instance-based (memory-based) learning methods form hypotheses directly from the training instances themselves, whereas model-based learning methods create general coherence by describing the implicit information of training data.

The key aspects of instance-based time series classifiers (e.g. k-nearest neighbor algorithm and its variations) are the representation methods and the (de)similarity measures. Time series representation techniques deal with the transformation of the high-dimensional time series data to an other feature space. The well-known representation methods are the Discrete Fourier Transformation (DFT) [8] , Singular Value Decomposition (SVD) [8] , Discrete Wavelet Transformation (DWT) [4] etc.

Their main functions are noise filtering and feature extraction. The similarity measures have more connections to the special attribute structure of time series, some of them are called elastic measures because they tolerate partial shifting or spreading of the time series values. Dynamic Time Warping (DTW) [11] and the edit distance based methods (Longest Common SubSequence(LCSS) [18] , Edit Distance on Real Sequence (EDR) [6] and Edit Distance with Penalty (EDP) [5]) are very efficient elastic similarity measures.

Typically, instance-based methods in time series classification provide efficient and accurate solutions [7] , but the selection of the appropriate representation method and the similarity measure require difficult cross-validation steps, more running time and expert knowledge. Extensive experimental comparision of representation and simirality measure can be read in [7].

Most model-based methods include some submethods to generate or predict the time series. For example, a Hidden Markov Model (HMM) can be built on time series with the same label, thus a time series in the test set is associated with the class of which HMM has the highest probability to generate the given time series.[17]. Similarly to this method, an other time series prediction method can be used in a classification algorithm, in which case the higher accuracy of the prediction method determines the labeling of the predicted time series. One member of the most popular and efficient time series prediction method family is the recurrent neural networks [10]. The classifiers based on these neural networks are accurate, however, their models are non-interpretable.

Typically, the instance-based method can not handle time series with different lengths, they require time series with equal lengths, whereas the prediction-based solutions can handle difference in the length of time series as well.

3 Classification of Time Series

3.1 Problem Definition

Time series Θ is a structured data, a finite vector of time value and observation vector pairs ($\Theta = \{< t_i, \boldsymbol{x_i} >\}_{i=1}^{T}$ where $\boldsymbol{x_i} =< x_i^1, x_i^2, \ldots, x_i^m >, x_i^j \in \Re$). The vector is ordered by the time parameter of its elements ($t_i \leq t_{i+1}$). In this paper we concentrate on equally sampled time series where $t_{i+1} - t_i$ equals $t_{j+1} - t_j$, and we assume that t_i equals i, so we can simplify Θ to $\{\boldsymbol{x_i}\}_{i=1}^{T}$ (i.e. a vector of observation vectors). Although the ShiftTree is also capable of classifying time series with multiple observations (i.e. multinomial time series), we concentrate on a simpler task in this paper: the $\boldsymbol{x_i}$ observation vector is replaced by x_i observation scalar. This type of structured data is also called value series in the literature. In the rest of this paper time series Θ refers to a series of x_i values.

In the classification task, we are given a training set of time series with class labels ($TR = \{< \Theta_n, L_n >\}_{n=1}^{N_{TR}}$) and a set of time series with unknown class labels. The task is to determine the value of the class labels of the elements of the latter set. The class labels get their values from a finite (and often small) set of values ($L_n \in CL = \{l_1, l_2, ..., l_{N_C}\}$). For the evaluation and comparison of different classifiers, a test set is used ($TE = \{< \Theta_n, L_n >\}_{n=1}^{N_{TE}}$). The TR and TE sets have no common elements.

There are many metrics for evaluating classifiers. In this paper we use accuracy. The classifier assigns a predicted class label $\hat{L_n}$ to the n^{th} series of the TE set. We define $\#hits$ as the number of correctly predicted class labels and the accuracy of the classifier as $Accuracy = \frac{\#hits}{N_{TE}} = \frac{\sum_{n=1}^{N_{TE}} Ind\{L_n = \hat{L_n}\}}{N_{TE}}$

3.2 Notation

- $TR \rightarrow$ The training set.
  - $N_{TR} \rightarrow$ The number of time series in the training set.
  - $TR[n] =< \Theta_n, L_n >\rightarrow$ The n^{th} element of the training set, a time series and class label pair.
  - $\Theta_n \rightarrow$ The n^{th} series in the training set.
  - $L_n \rightarrow$ The class label of the n^{th} series in the training set.
- $TE \rightarrow$ The test set. The meaning of N_{TE}, $TE[n]$, Θ_n^{te} and L_n^{te} are similar to N_{TR}, $TR[n]$, Θ_n and L_n. (The n^{th} series of the TR and TE sets are distinguished by the superscript ^{te}.)
- $CL \rightarrow$ The set of the possible class labels $\{l_1, l_2, ..., l_{N_C}\}$.
  - $N_C \rightarrow$ The number of different class labels.
- $\Theta \rightarrow$ A time series.
  - $\Theta[i] \rightarrow$ The i^{th} observation value of the time series Θ (i.e. x_i).
  - $T \rightarrow$ The length of the time series.

4 Concept

In this paper we propose a novel decision tree based algorithm called ShiftTree. In a node of an ordinary decision tree, the data set splitting criteria belongs to only a certain attribute x_i, but in the case of time series, adequate information is usually not in the same attribute x_i for each time series, as it may be found in an different attribute position for each time series. For example, the global maximum of time series would be an efficient splitting attribute, but their values can not be assigned to an exact position i in time series, to a certain attributes x_i, so the approach of vector based attribute representation is not adequate in this case.

In order to handle these problems, we assign a cursor (or eye) - denoted C - to every time series. The task of the cursor is to appoint an element of time series, and it can be interpreted as a position of its time series. This cursor can move back and forward on the time axis of the time series throughout the duration of our method. Initially, the cursors are set to the first position/attribute of the time series (It's assigned the attribute x_1, its value is 1). In our algorithm, every node of the decision tree has an operation of cursor, for example the cursor has to move to the next local maximum of time series. The result of this operation would be different for different time series, so this method has the possibility of implementing a time-elastic handling of time dimension. Attributes are computed dynamically using the position of the cursor, the value of the time series in that position and the surrounding values.

Every node of ShiftTree has an operation of computing the attribute, for example the attribute is the average of the values in the surroundings of the cursor with a radius of 5. In this way, each branch of the ShiftTree gives an interpretable description of the time series. An other important advantage of this approach is that the expert of the application field can define suitable operations to create a more accurate model for a specific problem.

The training of the novel decision tree model is based on selecting the appropriate operators (moving of cursor, attribute calculator) for each node. Our proposed training method is described in Section 5. The accuracy of the model depends on the set of usable operations from which a node can choose an appropriate one. One of our main goals was to create a general algorithm which is applicable to different fields, which we will display in Section 7, by using a basic set of operations to show that the accuracy of the models is satisfying on severely different time series classification problems. The accuracy can be further improved by using forest building methods. In section 6 we present two forest building methods based on boosting and cross-validation.

5 The ShiftTree Algorithm

5.1 The Structure of a ShiftTree Node

The main structure of our proposed algorithm is the ShiftTree, which is similar to the structure of decision tree algorithms: it is a binary tree with a root node,

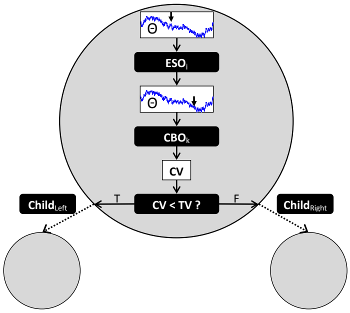


Fig. 1. Structure of a ShiftTree node

the leaf node contains the classification labels and decision points are associated with the not leaf nodes. As we mentioned above every node of the ShiftTree contains two operators: the first one describes how to move the cursor, the second one describes how to compute a dynamic attribute. The family of the first operator type is called EyeShifter operator (ESO) the second group is called ConditionBuilder operator (CBO).

Each node of the ShiftTree can be represented by the following structure of six elements $< ESO_j, CBO_k, TV, PLabel, Ch_L, Ch_R >$. ESO_j is an EyeShifter operator selected from a predefined set of ESOs ($j \in [1..N_{ESO}]$). An EyeShifter Operator ESO_j describes a shifting mode of a cursor on the given time series. It is important to understand that an ESO_j can shift the cursor on different time series to different positions. That is why the method can handle time series of different lengths in one classification task. CBO_k is a ConditionBuilder operator selected from a predefined set of CBOs ($k \in [1..N_{CBO}]$). CBO_k generates a dynamic attribute called Calculated Value (CV) using the position of the cursor C, the value of the time series in that position ($\Theta[C]$) and the nearby values. Ch_L and Ch_R are pointers to the left and right subtrees of the current node. If the node is a leaf then these two values are null. TV is called the threshold value. If the corresponding attribute of the time series is smaller than TV then the branch pointed by Ch_L will be the next one, in other cases the branch pointed by Ch_R will process the time series. The function $PLabel$ describes the labeling information in the node, $PLabel(l_i)$ returns with the confidence (probability) of the label l_i in the given node. The structure of a node is shown in Figure 1.

We present some simple operator examples for both ESO and CBO. The current position of the cursor is denoted by C, C_{new} is the new position of the cursor after applying the ESO, C_{prev} is the previous position of the cursor. The parameters of the operators are predefined, they are not changing during the learning process. We will show in Section 7 that a ShiftTree can be accurate using only this simple operator set.

Operator Examples

- ESONext($\mathit{\Delta} T$) → $C_{new} = min(C+\mathit{\Delta} T, T)$. Similar operator: ESOPrev($\mathit{\Delta} T$).
- ESONextMax(X) → $C_{new} = (i|\Theta[i] > max\{\Theta[i \pm 1]\}, C < i, \sum_{k=C+1}^{i-1} I_{\Theta[k]>max\{\Theta[k\pm 1]\}} = X - 1)$. Similar operator: ESOPrevMax(X), ESONextMin(X), ESOPrevMin(X).
- ESOMax(global) → $C_{new} = argmax_i(\Theta[i])$. Similar operator: ESOMin(global).
- ESOMax(sofar) → $C_{new} = argmax_{i=1...C}(\Theta[i])$. Similar operator: ESOMin(sofar).
- ESOClosestMax → $C_{new} = min_{|C-i|}(i|\Theta[i] > \Theta[i-1], \Theta[i] \geq \Theta[i+1])$ Similar operator: ESOClosestMin.
- ESOGreaterMax → $C_{new} = argmax(\Theta[ESONextMax(1)], \Theta[ESOPrevMax(1)])$ Similar operators: ESOGreaterMin, ESOLesserMax, ESOLesserMin.
- ESOMaxInNextInterval($\mathit{\Delta} T$) → $C_{new} = argmax_{i=0...\mathit{\Delta} T}(\Theta[C+i])$. Similar operators: ESOMaxInPrevInterval($\mathit{\Delta} T$), ESOMinInNextInterval($\mathit{\Delta} T$), ESOMinInPrevInterval($\mathit{\Delta} T$).
- ComplexESO → This operator is a vector of two or more ESOs. It moves the cursor by its first ESO then by its second ESO and so on.
- CBOSimple → $CV = \Theta[C]$
- CBONormal(μ,σ,X) → $CV = average\{\exp^{-\frac{\mu^2}{2\sigma^2}} \Theta[C], \exp^{-\frac{(|i|-\mu)^2}{2\sigma^2}} \Theta[C \pm i]|i = 1 \dots X\}$
- CBOExp(λ,X) → $CV = average\{\lambda\Theta[C], \exp^{-\lambda|i|} \Theta[C \pm i]|i = 1 \dots X\}$
- CBOLinear(X) → $CV = average\{\Theta[C], \frac{1}{|i|}\Theta[C \pm i]|i = 1 \dots X\}$
- CBOAVG(X) → $CV = average\{\Theta[C], \Theta[C \pm i]|i = 1 \dots X\}$
- CBODeltaT(norm/abs) → $CV = C - C_{prev}$ or $CV = |C - C_{prev}|$
- CBOTimeSensitive(norm/abs) → $CV = \frac{\Theta[C]}{C-C_{prev}}$ or $CV = \frac{\Theta[C]}{|C-C_{prev}|}$
- CBO[Average/Variance](sofar/delta) → Returns the average/variance of the values $\{\Theta[1], ..., \Theta[C]\}$ or $\{\Theta[C_{prev}], ..., \Theta[C]\}$
- CBO[Max/Min][AVG/VAR/Count](sofar/delta) → Returns the average/variance/number of the local maximums/minimums in the subseries of $\{\Theta[1], ..., \Theta[C]\}$ or $\{\Theta[C_{prev}], ..., \Theta[C]\}$.
- CBOMedian(B, F) → $CV = median\{\Theta[C-B], \Theta[C-B+1], ..., \Theta[C], ..., \Theta[C+F-1]\Theta[C+F]\}$

5.2 Classification Process

The ShiftTree's classification process for time series Θ can be written by the next recursive process (see Algorithm 5.1). The input of the first call has to be a ShiftTree represented by its root node R and the unlabeled time series Θ and the initial cursor position ($C = 0$).

The function $ShiftCursor(ESO_j, \Theta, C)$ shifts the cursor of time series from position C to a new one by applying EyeShifter operator ESO_j, the function $CalculateValue(CBO_k, \Theta, C)$ calculates a value over time series Θ by using ConditionBuilder operator CBO_k and the cursor position C.

Algorithm 5.1. Labeling process of the ShiftTree

```
Input: node $R$, time series $\Theta$, cursor $C$
Output: label $L \in [l_1, ..., l_{N_L}]$ for time series $\Theta$
procedure ShiftTreeLabel($R$, $\Theta$, $C$)
  $R \rightarrow < ESO_j, CBO_k, TV, PLabel, Ch_L, Ch_R >$
  if $R$ is not a leaf then
      $C_{new} \leftarrow$ ShiftCursor($ESO_j$, $\Theta$, $C$)
      $CV \leftarrow$ CalculateValue($CBO_k$, $\Theta$, $C_{new}$)
      if $CV < TV$ then
          $L \leftarrow$ ShiftTreeLabel($Ch_L$, $\Theta$, $C_{new}$)
      else
          $L \leftarrow$ ShiftTreeLabel($Ch_R$, $\Theta$, $C_{new}$)
      end if
  else
      $L \leftarrow argmax_{l_i} PLabel(l_i)$, $l_i \in [l_1, ..., l_{N_L}]$
  end if
  return $L$
end procedure
```

5.3 Training Process

The learning process of the ShiftTree is more complicated (see Algorithm 5.2). The process is defined by the generation method of only one ShiftTree node, because the training method can be defined as a recursive algorithm. In this case the input is a training set $TR = \{< \Theta_n, L_n >\}_{n=1}^{N_{TR}}$. The output of process is a subtree of the ShiftTree represented by its root node R. The process tries to find an accurate ESO_j, CBO_k and TV setting, because this triple determines a splitting criteria in a given node. The algorithm selects the best splitting criteria by minimizing the entropy of the child nodes. Note that this is the same as maximizing the information gain of the splitting. The entropy is defined as follows:

$$Ent(TR_L, TR_R) = - \sum_{X \in [L,R]} \frac{N_X}{N} \sum_{i=1}^{N_C} (P_{Xi} * \log_2 P_{Xi}) \qquad (1)$$

TR_L and TR_R are the two sets of time series label pairs. N , N_L and N_R are the number of time series in $TR_L \bigcup TR_R$, TR_L and TR_R. P_{Li} and P_{Ri} are the relative frequency of the label l_i in TR_L and TR_R. N_C is the number of class labels.

The function $StoppingCriteria(PLabels)$ return true if $PLabels(l_i) = 1$ for a class label value $l_i \in CL$. We experimented with other stopping criteria but this one gave the best results on the benchmark datasets. If the node is not a leaf, every ESO_j CBO_k pairs are examined by the training algorithm. $ShiftCursor(ESO_j, \Theta, C)$ and $CalculateValue(CBO_k, \Theta, C)$ are the same as they were in algorithm 5.1. When the CVs are calculated for all time series in TR, every sensible threshold value is examined. The easiest way to do this is to sort the CVs and set TV to be the mean of every two adjacent CVs

Algorithm 5.2. Recursive learning method of ShiftTree

```
Input: a set of labeled time series $TR = \{< \Theta_n, L_n >\}_{n=1}^{N_{TR}}$ and their cursors $\{C_n\}_{n=1}^{N_{TR}}$
Output: Node $R$ that represents the newly created subtree of the ShiftTree
procedure BuildShiftTree($TR$,$\{C_n\}_{n=1}^{N_{TR}}$)
New node $R$
for all $l_i \in CL$ do
    $PLabels(l_i) \leftarrow \frac{|\{n|L_n=l_i\}|}{N_{TR}}$
end for
if StoppingCriteria($PLabels$) = true then
    $R \leftarrow leaf$
else
    for all $ESO_j \in ESO$ do
        for all $CBO_k \in CBO$ do
            for $n = 1..N_{TR}$ do
                $C_{new}^{j,n} \leftarrow$ ShiftCursor($ESO_j$,$\Theta_n$,$C_n$)
                $CV^{j,k,n} \leftarrow$ CalculateValue($CBO_k$,$\Theta_n$,$C_{new}^{j,n}$)
            end for
            $[CV^{j,k,1}, CV^{j,k,2}, \ldots, CV^{j,k,N_{TR}}] \leftarrow$ Sort($[CV_1^{j,k}, CV_2^{j,k}, \ldots, CV_{N_{TR}}^{j,k}]$)
            for $m = 1..N_{TR} - 1$ do
                $TV^{j,k,m} \leftarrow (CV_m^{j,k} + CV_{m+1}^{j,k})/2$
                $TR_L^{j,k,m} \leftarrow \{\Theta_n | CV^{j,k,n} < TV\}$
                $TR_R^{j,k,m} \leftarrow \{\Theta_n | CV^{j,k,n} \geq TV\}$
                $E^{j,k,m} \leftarrow$ Ent($TR_L^{j,k,n}$, $TR_R^{j,k,n}$)
            end for
        end for
    end for
    $< j'_q, k'_q, m'_q >_{q=1}^{Q} = \{< j,k,m > | E^{j,k,m} = min_{j,k,m} E^{j,k,m}\}$
    $j', k', m' = argmax_{j'_q,k'_q,m'_q} H_1(\{CV_n^{j'_q,k'_q}\}_{n=1}^{N_{TR}}, TV^{j'_q,k'_q,m'_q})$
    for $n = 1..N_{TR}$ do
        $C_n \leftarrow C_{new}^{j',n}$
    end for
    $TR_L \leftarrow TR_L^{j',k',m'}$
    $TR_R \leftarrow TR_R^{j',k',m'}$
    $Cursors_L \leftarrow \{C_n | \Theta_n \in TR_L\}$
    $Cursors_R \leftarrow \{C_n | \Theta_n \in TR_R\}$
    $Ch_L \leftarrow$ BuildShiftTree($TR_L$,$Cursors_L$)
    $Ch_R \leftarrow$ BuildShiftTree($TR_R$,$Cursors_R$)
    $R \leftarrow < ESO_{j'}, CBO_{k'}, TV^{j',k',m'}, PLabels, Ch_L, Ch_R >$
end if
return $R$
end procedure
```

(line 14 - 19), because by doing so we examine every possible splitting of the TR set by the current dynamical attribute. Lines 23 - 24 select the best splitting. As we mentioned above the algorithm selects the splitting which minimizes the entropy of the child nodes. In case of small training datasets, there may be several $< j, k, m >$ triplets that minimizes the expression in line 23. One should think that selecting one from equally good triplets is meaningless but our experiments have shown that the selection can significantly affect the accuracy of the model. The training set contains no trivial information to distinct these triplets properly so we rely on heuristics. We defined two similar heuristics that were based on the fact that the $CV^{j,k,n}$ values should be as far away from TV as possible. It can be assumed that a member of the test set has lower probability of ending up on the wrong side of the splitting if the CVs of the element of TR_L and TR_R are more distinct. We also had to use some kind of normalization because the CBOs might work in different range. This can be achieved in many ways, we found the following heuristic satisfying:

$$H_1(\{CV_n^{j,k}\}_{n=1}^{N_{TR}}, TV^{j,k,m}) = \frac{CV_{m+1}^{j,k} - CV_m^{j,k}}{CV_{N_{TR}}^{j,k} - CV_1^{j,k}} \quad (2)$$

If a triplet $< j, k, m >$ maximizes formula (2), then the CVs of the elements of $TR_L^{j,k,m}$ and $TR_R^{j,k,m}$ are rather distinct from each other. There may be some nodes where more than one $< j', k', m' >$ triplets minimize (1) and maximize (2), but those nodes don't seem to be significant as they usually have a TR set of only a couple of time series. In that case the first appropriate triplet is selected.

At the end of the process (lines 25 - 34) we set cursor C to its new position, split the TR set into two sets (TR_L, TR_R), create the child nodes using the same process on the elements of TR_L and TR_R. Note that Algorithm 5.2 is for demonstrative purposes only, for example collecting all possible TVs is not optimal and there are other issues one should consider when implementing this algorithm. Computational complexity may seem to be high, but a semi-optimal implementation of the ShiftTree was much faster than 1-NN using Eucledian distance or DTW.

5.4 About Interpretability

Interpretability is often underestimated but it can be of great importance in practical applications as most of the users do not trust machine learning algorithms unconditionally. If a model is interpretable, one can check if it learned an unimportant feature of the data or noise. An other advantage of interpretability - besides gaining trust of the users - is that we can learn the previously unknown properties of a problem. If a ShiftTree model is analyzed, special decision scenarios can be created by following different branches of the tree. If the ESOs and CBOs are simple interpretable operations, the experts can be understood deeper correspondences by considering the cursor scenarios.

6 Forest Methods for ShiftTree

It is a common method to create different models for a given classification problem and then combine the output of those model in order to achieve improved accuracy. The models can be the results of one or many algorithms. Building and combining only decisions trees is often called forest building. In this section we briefly introduce two forest approaches which we used to improve the accuracy of ShiftTree models.

6.1 Boosting

One of the most common methods for combining is boosting [9]. This iterative method assigns weights to the elements of the training set, trains a model and assigns a weight to the model, based on its weighted classification error. The weights assigned to the elements of the training set is also modified in a way that the weights of the correctly classified elements decrease and the weights of the rest of the elements increase. The combined output is a weighted vote on the label. The widely used AdaBoost [9] technique has a precondition that the weighted classification error of the model must be lesser than 50%. This is the same error rate as the error of random guessing on a classification problem of 2 classes. Since we tested our algorithm on some problems which have many classes (up to 50), we selected an other boosting technique that has a less strict precondition. This boosting technique is called SAMME [19] and requires an error rate lesser than $100\% - \frac{1}{N_C}$ which is the same as the error of random guessing on a problem with N_C classes. This method assigns the weight W_m to the model and increases the weight of the wrongly classified elements. The weights of the correctly classified elements are not updated but after the update the sum of all weight is normalized to 1. Like AdaBoost, this method also stops when the error of the model is 0%. We had to solve the problem that the ShiftTree often creates a model that fits to the entire training set (in other words, the model classifies every single element of the training set correctly). We experimented with many pruning techniques. Every one of them decreased the accuracy of the models on the test set. In the case of low accuracy the combined models are better than an accurate single model. We used the common chi-square post-pruning [16].

6.2 XV Method

This combination technique receives its name after cross-validation. Only a part of the training set is used for training, the other part serves as a validation set (VA) on which we measure the predicted accuracy of the model. We assign the predicted accuracy to the model as the weight of the model. The combined output is a weighted vote on the label, so this method implements a simple ensambled method over ShiftTree construction. The two parameters of this method are the iteration number M and the ratio of the sizes of the VA and (original) TR sets.

7 Numerical Results

In this section we present the results of the ShiftTree on some datasets and compare them to the accuracy of widely used instance-based methods. We examined both the basic algorithm and the forest building methods. We also did blind tests that took place in a contest environment and compared our results to the results of the participants of that competition.

7.1 Datasets and Testing Environment

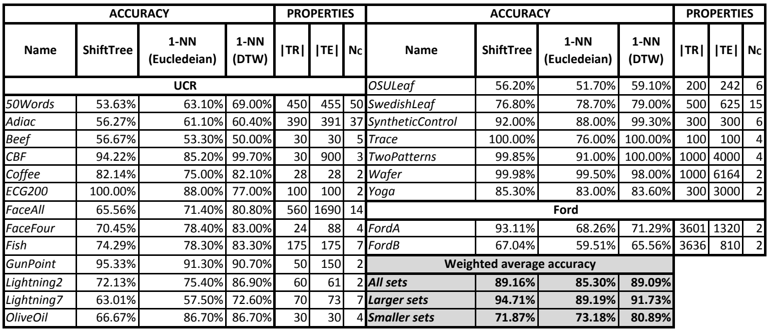
We used three databases for the evaluation. The first database is one of the largest publicly available time series databases [13] often used as a benchmark database. It will be referred to as the UCR database. This database consists of 20 classification problems (datasets). Each set is originally divided into a training and a test set. We used these original splits. About half of the training sets in this database are small. While it is important to check the results of ShiftTree on these classification problems too, we do not expect high accuracy on these problems as ShiftTree is a model-based algorithm. The second database consists of the 2 datasets of the Ford Classification Challenge [1]. These datasets were originally divided into 3 sets (training, validation, test). We merged the training and validation sets into a training set by both datasets and used the test set for testing. This database will be referred to the Ford database. These two databases were used for the normal testing of our algorithm. The third database comprises the data of the SIGKDD2007 Time Series Classification Challenge [12]. This database will be referred to the TSC database. This database consists of 20 classification problems and the properties of the datasets are similar to the properties of the UCR database. We used the TSC data for the blind tests. The properties of the datasets can be seen in figure 2.

As one of our goals was to create a generally accurate algorithm, we decided to use the same operators for every problem. The description of these operators is in section 5. The parameters of the operators were also the same by all problems. The value of the parameters were determined based on the minimal and maximal length of time series of all datasets. Some operators were used more than once (with different parameterization). A total of 130 ESOs and 48 CBOs were used, so 6240 dynamic attributes were considered in each node.

7.2 Results of the Basic ShiftTree

Figure 2 shows the accuracy values for the problems of the UCR and Ford databases. The weighted accuracy is the number of correctly classified series in all test sets divided by the number of test samples (i.e. the weights are the sizes of the test sets). The accuracy of the widely used 1-NN algorithm with both Euclidian distance and DTW are also shown. We used the results reported on [13] for 1-NN.

ShiftTree has the highest overall accuracy of the 3 algorithms, but the ranking of the algorithms at different problems vary. As it is expected from a model based



ACCURACY				PROPERTIES		
Name	ShiftTree	1-NN (Eucledeian)	1-NN (DTW)	|TR|	|TE|	Nc
UCR						
50Words	53.63%	63.10%	69.00%	450	455	50
Adiac	56.27%	61.10%	60.40%	390	391	37
Beef	56.67%	53.30%	50.00%	30	30	5
CBF	94.22%	85.20%	99.70%	30	900	3
Coffee	82.14%	75.00%	82.10%	28	28	2
ECG200	100.00%	88.00%	77.00%	100	100	2
FaceAll	65.56%	71.40%	80.80%	560	1690	14
FaceFour	70.45%	78.40%	83.00%	24	88	4
Fish	74.29%	78.30%	83.30%	175	175	7
GunPoint	95.33%	91.30%	90.70%	50	150	2
Lightning2	72.13%	75.40%	86.90%	60	61	2
Lightning7	63.01%	57.50%	72.60%	70	73	7
OliveOil	66.67%	86.70%	86.70%	30	30	4
OSULeaf	56.20%	51.70%	59.10%	200	242	6
SwedishLeaf	76.80%	78.70%	79.00%	500	625	15
SyntheticControl	92.00%	88.00%	99.30%	300	300	6
Trace	100.00%	76.00%	100.00%	100	100	4
TwoPatterns	99.85%	91.00%	100.00%	1000	4000	4
Wafer	99.98%	99.50%	98.00%	1000	6164	2
Yoga	85.30%	83.00%	83.60%	300	3000	2
Ford						
FordA	93.11%	68.26%	71.29%	3601	1320	2
FordB	67.04%	59.51%	65.56%	3636	810	2
Weighted average accuracy						
All sets	**89.16%**	**85.30%**	**89.09%**			
Larger sets	**94.71%**	**89.19%**	**91.73%**			
Smaller sets	**71.87%**	**73.18%**	**80.89%**			

Fig. 2. Results of the basic ShiftTree, 1-NN (Euclidian) and 1-NN (DTW). Also contains some basic properties of the datasets.

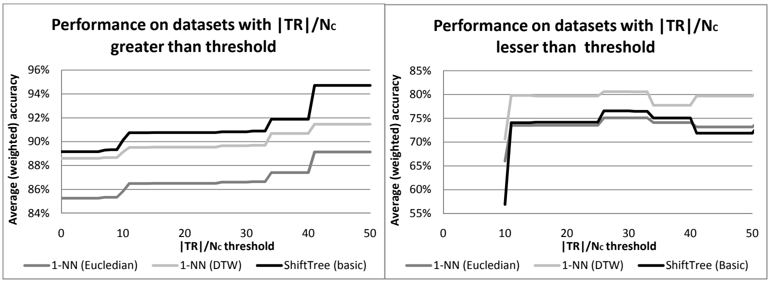


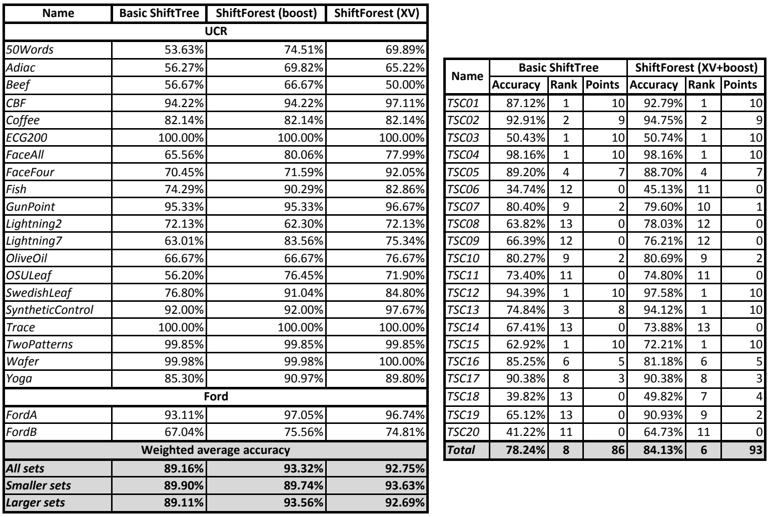
Fig. 3. Results of the basic ShiftTree, 1-NN (Euclidian) and 1-NN (DTW) on datasets with greater and lesser $\frac{|TR|}{N_C}$ values than a moving threshold

method, ShiftTree is less effective on smaller datasets. The average accuracy of the algorithms on smaller/larger datasets are shown on figure 2. We considered datasets "smaller" if the average number of instances per a class in the training set ($\frac{|TR|}{N_C}$) is lesser than than a threshold value (40). ShiftTree outperforms the neighbor based algorithms if it is provided with enough samples of every class, but loses to them if there are only a few samples available. Figure 3 shows the accuracy of all three algorithms on both "smaller" and "larger" datasets using different threshold values. At a threshold value Th the datasets of the UCR and Ford databases were divided into two groups: the ones with lesser $\frac{|TR|}{N_C}$ value than Th were considered "smaller" an the others "larger". We computed the weighted accuracy (all correctly classified test samples divided by all test samples) for both groups. By increasing the threshold, ShiftTree gains greater advantage on 1-NN using the "larger" datasets. By lowering the threshold the gap between ShiftTree and 1-NN widens on "smaller" datasets. This proves our assumption that our model based approach performs well mostly on larger datasets.

The average running time of the basic ShiftTree (training) algorithm was 6.48 seconds per dataset in case of UCR dataset collection (minimum 0,144 sec - CBF; maximum 33,99 sec - 50Words). The running times on the larger FordA and FordB datasets were 200.5 and 173.5 seconds.

7.3 Results of the Forest Methods

We made several experiments with the forest building techniques (described in 6). We found that the accuracy of boosting increases continuously as the number of iterations is increased. We finally set the number of iterations to 100 which is an acceptable trade-off between speed and accuracy. The significance level of pruning was set to 0.01% (strict pruning). Increasing number of iterations by the XV method did not really affect the accuracy above 20 so we used 20 as the M parameter. The XV method has another parameter: the size of the validation set (S). If it is set too low then the variance of the predicted accuracy values will be high and these accuracies are useless as model weights (because they are inaccurate). If we set S too high, then the ShiftTree models will be inaccurate as there are only a few samples for the training. We found 30% to be the optimal value for S. The results of both methods (using the optimal parameters) and the basic method can be seen in Figure 4. Boosting seems to be the better forest building technique. But if we look closer, the XV greatly outperforms boosting on some datasets. The common property of these datasets is that their training set is small. If we examine the average accuracy on datasets having less than 70 samples for training (smaller sets), even the basic method outperforms



Name	Basic ShiftTree	ShiftForest (boost)	ShiftForest (XV)
UCR			
50Words	53.63%	74.51%	69.89%
Adiac	56.27%	69.82%	65.22%
Beef	56.67%	66.67%	50.00%
CBF	94.22%	94.22%	97.11%
Coffee	82.14%	82.14%	82.14%
ECG200	100.00%	100.00%	100.00%
FaceAll	65.56%	80.06%	77.99%
FaceFour	70.45%	71.59%	92.05%
Fish	74.29%	90.29%	82.86%
GunPoint	95.33%	95.33%	96.67%
Lightning2	72.13%	62.30%	72.13%
Lightning7	63.01%	83.56%	75.34%
OliveOil	66.67%	66.67%	76.67%
OSULeaf	56.20%	76.45%	71.90%
SwedishLeaf	76.80%	91.04%	84.80%
SyntheticControl	92.00%	92.00%	97.67%
Trace	100.00%	100.00%	100.00%
TwoPatterns	99.85%	99.85%	99.85%
Wafer	99.98%	99.98%	100.00%
Yoga	85.30%	90.97%	89.80%
Ford			
FordA	93.11%	97.05%	96.74%
FordB	67.04%	75.56%	74.81%
Weighted average accuracy			
All sets	**89.16%**	**93.32%**	**92.75%**
Smaller sets	**89.90%**	**89.74%**	**93.63%**
Larger sets	**89.11%**	**93.56%**	**92.69%**

Name	Basic ShiftTree			ShiftForest (XV+boost)		
	Accuracy	Rank	Points	Accuracy	Rank	Points
TSC01	87.12%	1	10	92.79%	1	10
TSC02	92.91%	2	9	94.75%	2	9
TSC03	50.43%	1	10	50.74%	1	10
TSC04	98.16%	1	10	98.16%	1	10
TSC05	89.20%	4	7	88.70%	4	7
TSC06	34.74%	12	0	45.13%	11	0
TSC07	80.40%	9	2	79.60%	10	1
TSC08	63.82%	13	0	78.03%	12	0
TSC09	66.39%	12	0	76.21%	12	0
TSC10	80.27%	9	2	80.69%	9	2
TSC11	73.40%	11	0	74.80%	11	0
TSC12	94.39%	1	10	97.58%	1	10
TSC13	74.84%	3	8	94.12%	1	10
TSC14	67.41%	13	0	73.88%	13	0
TSC15	62.92%	1	10	72.21%	1	10
TSC16	85.25%	6	5	81.18%	6	5
TSC17	90.38%	8	3	90.38%	8	3
TSC18	39.82%	13	0	49.82%	7	4
TSC19	65.12%	13	0	90.93%	9	2
TSC20	41.22%	11	0	64.73%	11	0
Total	**78.24%**	**8**	**86**	**84.13%**	**6**	**93**

Fig. 4. LEFT: Results of the basic ShiftTree, boosting used 100 iterations and XV used 20 iterations and 30% of the training set as the validation set. A dataset is considered "smaller" if its training set contains less than 70 series.
RIGHT: Results of the basic ShiftTree and the ShiftForest on the blind test.

boosting by a bit. The reason for this is that even the pruned ShiftTree model fits perfectly to the small set of training data therefore the boosting stops after one iteration. Thus we basically get the original ShiftTree algorithm back. XV uses different training sets that insures to build different models. And by averaging those models, improved accuracy can be achieved even on smaller datasets. But XV is much more simpler than boosting so if enough training data is available, boosting will surpasses XV. We found that "enough" means 70 training samples by the UCR database. By using the appropriate method for all datasets, an average accuracy of 93.57% can be achieved.

7.4 Results of the Blind Tests

As we mentioned above, we used the datasets of the SIGKDD2007 Time Series Challenge to evaluate our algorithm in a blind test. We did one test for the basic algorithm and one for the best forest method. The parameterization of the basic method is the same as in the previous subsections. The parameterization of the forest method is the same as the best parameterization for the UCR datasets: on datasets having a training set of less than 70 examples, we used XV ($M = 20$, $S = 30\%$), on the rest we used boosting with 100 as the number of iterations. We calculated the points for ShiftTree as it had taken part in the competition: 10 point for a 1^{st} place, 9 for a 2^{nd} and so on. We also modified the points of the other participants, if the ShiftTree surpassed the accuracy of their algorithms. The two methods took part in two separate tests (i.e. they were not competing each other). Figure 4 shows the results.

Out of 12+1 participants basic ShiftTree gained a total rank of 8. We consider this to be a great success as the operators/parameters were not optimized and we assume that most of the participants used blended algorithms. Interestingly, ShiftTree ranked 1^{st} place 5 times out of 20 which is the same amount as the overall winner has. We think that the reason for this is that ShiftTree is accurate on datasets with greater average training examples per class values. The ShiftTree forest improves the overall results: it gains one more 1^{st} place as the overall winner loses one and moves forward to the 6^{th} place of the challenge.

8 Conclusion

We proposed a new model-based time series classifier called ShiftTree, which is an important advance in this research field because of its unique benefits: thanks to its model-base approach, the decision tree based model can be interpretable, this property is infrequent in this data mining domain, where the instance-based algorithms are overrepresented.

The key aspect of time series classification is the handling of correspondences of time dimension. Instead of an elastic time approach, we propose a novel attribute indexing technique: a cursor is assigned to each instance of the time series dataset. By determining the cursor operators, our algorithm can classify with high accuracy. The supervised learning method of the ShiftTree is an extension of the decision tree building methods, where the cursor operator and attribute calculator modes are designated beyond the splitting value.

The numerical results of 22 datasets of a benchmark collection show that our algorithm has similar accuracy to other techniques, moreover its accuracy exceeds the best instance-based algorithms on some datasets. As it is typical of model-based algorithms, the ShiftTree algorithm can work more efficiently than the instance-based methods when the size of training dataset is larger. The proposed forest extension of ShiftTree, the cross-validation based and boosting techniques are to improve the accuracy level of ShiftTree. The efficiency of the algorithm can be further enhanced by defining domain specific cursor operators and attribute calculators, where the specific characteristics of the dataset are also used.

Although the efficiency of our algorithm is significant in some cases, its importance lies in the fact, that by novel cursor-based attribute indexing, it can solve some classification problems which can not be answered by the other model-based or instance-based methods. We think that this attribute indexing technique is promising and it can be the base of a novel algorithm family in the future in the field of time series and spatial data mining.

References

1. Abou-Nasr, M., Feldkamp, L.: Ford Classification Challange (2008), `http://home.comcast.net/~nn_classification/`
2. Acir, N.: Classification of ecg beats by using a fast least square support vector machines with a dynamic programming feature selection algorithm. Neural Computing and Applications 14(4), 299–309 (2005)
3. Azoff, E.M.: Neural Network Time Series: Forecasting of Financial Markets. Wiley, Chichester (1994)
4. Pong Chan, K., Chee Fu, A.W.: Efficient time series matching by wavelets. In: ICDE (1999)
5. Chen, L., Ng, R.: On the marriage of lp-norms and edit distance. In: VLDB (2004)
6. Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: SIGMOD Conference (2005)
7. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: Experimental comparison of representations and distance measures. In: VLDB (2008)
8. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: SIGMOD Conference Proceedings (1994)
9. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences (55), 119–139 (1997)
10. Jager, H.: The echo state approach to analysing and training recurrent neural networks. GMD Report 148 8, 1–42 (August 2001)
11. Keogh, E., Ratanamahatana, C.A.: Exact indexing of dynamic time warping. Knowl. Inf. Syst. 7(3) (2005)
12. Keogh, E., Shelton, C., Moerchen, F.: Workshop and Challenge on Time Series Classification at SIGKDD 2007 (2007), `http://www.cs.ucr.edu/~eamonn/SIGKDD2007TimeSeries.html`

13. Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.A.: The UCR Time Series Classification/Clustering Homepage (2006),
http://www.cs.ucr.edu/~eamonn/time_series_data/
14. Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding motifs in time series. In: 2nd Workshop on Temporal Data Mining (KDD 2002), pp. 53–68 (2002)
15. Prekopcsak, Z.: Accelerometer based real-time gesture recognition. In: POSTER 2008: Proceedings of the 12th International Student Conference on Electrical Engineering (May 2008)
16. Quinlan, J.R.: Induction of decision trees. In: Readings in Machine Learning. Morgan Kaufmann, San Francisco (1990)
17. Vlachos, M., Kollios, G., Gunopulos, D.: Discovering similar multidimensional trajectories. Data Engineering (August 2002)
18. Vlachos, M., Gunopulos, D., Kollios, G.: Discovering similar multidimensional trajectories. In: ICDE (2002)
19. Zhu, J., Rosset, S., Zou, H., Hastie, T.: Multi-class adaboost. Statistics and Its Interface 2, 349–360 (2009)