FEARS: a FEature And Representation Selection approach for time series classification

Alexis Bondu ALEXIS.BONDU@ORANGE.COM
Orange Labs, Paris, France

Dominique Gay DOMINIQUE.GAY@UNIV-REUNION.FR
LIM-EA2525, Université de La Réunion, France

Vincent Lemaire VINCENT.LEMAIRE@ORANGE.COM
Orange Labs, Lannion, France

Marc Boullé MARC.BOULLE@ORANGE.COM
Orange Labs, Lannion, France

Eole Cervenka EOLECVK@GMAIL.COM
Boston Consulting Group, San Francisco, United States

Editors: Wee Sun Lee and Taiji Suzuki

Abstract

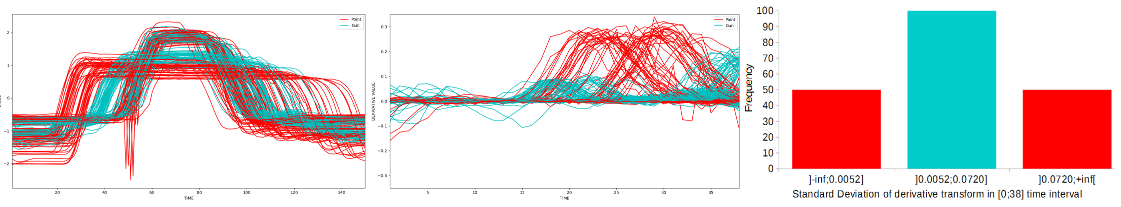
This paper presents a method which extracts informative features while selecting simultaneously adequate representations for Time Series Classification. This method simultaneously (i) selects alternative representations, such as derivatives, cumulative integrals, power spectrum ... (ii) and extracts informative features (via automatic variable construction) from the selected set of representations. The suggested approach is decomposed in three steps: (i) the original time series are transformed into several representations which are stored as relational data; (ii) then, a regularized propositionalisation method is applied in order to generate informative aggregate features; (iii) finally, a selective Naive Bayes classifier is learned from the outcoming feature-value data table. The previous steps are repeated by a forward backward selection algorithm in order to select the most informative subset of representations. The suggested approach proves to be highly competitive when compared with state-of-the-art methods while extracting interpretable features. Furthermore, the suggested approach is almost parameter free and only requires few hardware resources.

Keywords: Time Series Classification, Multiple Representations, Propositionalisation, Representation Selection

1. Introduction

Time series analysis is widely used in many application areas, notably due to the increasing use of sensors in industry, health, meteorology, and IoT domain. Time series research deals with a large range of learning tasks such as forecasting, compression, clustering, etc. This paper addresses the problem of time series classification (TSC). For an incoming univariate time series denoted by $\tau_i = \langle(t_1, x_1), (t_2, x_2), \ldots, (t_m, x_m)\rangle$, where x_k is the value of the series at time t_k, the goal is to predict the value of a categorical target variable given a set of labeled time series. The amount of TSC literature is substantial: many TSC methods have been suggested in recent years, and progress has mainly focused on improving the accuracy of classifiers (Bagnall et al. (2017)). Besides distance-based (Wang et al. (2013)), interval-based (Baydogan et al. (2013)), shapelet-based (Mueen et al.

(2011)), dictionary-based (Schäfer (2015); Schäfer and Leser (2017)), deep learning based (Fawaz et al. (2019)) and ensemble methods (Lines and Bagnall (2015)), a consensus has emerged from the TSC community that transforming time series from the time domain to an alternative data space is one of the best catalyst for accuracy improvement. Combining several classifiers that learn from several representations in an ensemble method is the way COTE (Bagnall et al. (2015)) and HIVE-COTE (Lines et al. (2018)) work and perform the best accuracy results to date on the UCR/UEA TSC benchmark data sets (Bagnall et al. (2018)). However, as observed by the authors, it is at the expense of very high computational cost. Moreover, the ensemble methods embed various k-NNs, Naïve Bayes, SVMs, decision tree ensembles are inherently deprived of interpretability perspectives[1]. According to the authors, HIVE-COTE (Lines et al. (2018)) provides almost optimal results in terms of accuracy, and now, the researchers' objective should be to develop approaches with comparable accuracy, and better on other criteria such as scalability, intelligibility, automation etc. This motivated us to suggest FEARS: a new time-efficient and accurate TSC method. This method extracts informative features while selecting simultaneously adequate representations for Time Series Classification, such as derivatives, cumulative integrals, and power spectrum.



(a) (b) (c)

Figure 1: GunPoint dataset in (a) original time, (b) derivative representation with a focus on time interval $[0; 38]$, (c) the GunDraw/Point class distribution w.r.t. the value of standard deviation of the series in derivative representation in $[0; 38]$

As a motivating example, Figure 1 shows an example of a very discriminative feature extracted by our approach from the famous GunPoint data set. This example relates the potential discriminative power of our approach for extracting features from alternative representations. Figure 1:a represents the time series in the original time domaine. At first sight, the two classes seem difficult to discriminate against, as the red and green time series are confusedly intertwined. Figure 1:b shows the derivative representation of these time series on the time interval $[0; 38]$. Visually, the derivative appears to be a more suitable representation for discriminating between classes. On this data set, the suggested approach has extracted a very informative aggregate feature which is the standard deviation values of derivative series, selected in the time interval $[0; 38]$. Three class-discriminative intervals are identified by the used classifier. All the time series belonging to the "GunDraw" class have a standard deviation between $v_1 = 5.2 \times 10^{-3}$ and $v_2 = 7.2 \times 10^{-2}$, while the other values correspond to the "Point" class. Here, our approach extracts an aggregate feature which holds valuable information about the target variable when discretized into three intervals. A straightforward interpretation might highlight the starting time at which an individual begins to raise his arm. It seems that individuals from "Point" class show two patterns: *(a)*, lifting their arm after time-stamp 38, thus having very low speed deviation in the considered time interval (with $v \leq v_1$)

1. The recent General Data Protection Regulation (GDPR) suggests that one should be able to give explanations of the decisions proposed by automated processing.

and *(b)*, reaching the top of lifting movement before time-stamp 38, thus having higher speed deviation (with $v \geq v_1$) than the rest of individuals. Finally, individuals belonging to the "GunDraw" class, which point a gun, show an in between pattern.

The suggested approach generalizes the underlying concepts of the illustrative example as follows: *(i)*, firstly, we transform the original time series into multiple representations which are stored in secondary tables as in relational data scheme; *(ii)*, then, informative and robust descriptors are extracted from relational data, using a regularized Bayesian propositionalisation method; *(iii)*, thirdly, a selective Naïve Bayes classifier is trained on the obtained flattened data; *(iv)*, in a looping scheme, these steps are repeated by a forward backward selection algorithm in order to find the most informative subset of representations.

The rest of the paper is organized as follows. Section 2 presents the related work organized within two research fields, time series classification and propositionalisation. Section 3 provides the main concepts of the FEARS approach. We present extensive comparative experiments on 85 public datasets from various application areas in Section 4 before concluding and opening future perspectives in Section 5.

2. Related work

TSC over multiple representations - As pointed in (Bagnall et al. (2017)), the diversity of TSC literature might be grouped w.r.t. the underlying discriminatory features that the various methods are looking for. Readers may refer to (Bagnall et al. (2017)) for a well-structured survey on recent TSC methods. A transversal and effective way to tackle with TSC is to pre-process the original time series by transforming them into alternative representations. In (Bagnall et al. (2012)), three transformations are performed (using power spectrum, autocorrelation and principal components), then a classifier is built on each transformed data and the classifiers are combined using various weighted voting schemes. In (Bagnall et al. (2015)), the authors extend the previous idea by integrating two supplementary autocorrelation-based transforms, a shapelet transform and then combining 35 classifiers. As embedded classifiers hold various biases, the whole ensemble is able to find various discriminatory features in the considered representations thus improving accuracy results. Recently, (Lines et al. (2018)) suggested a probabilistic hierarchical combination of five constituent ensembles and obtained the best accuracy results to date on UEA/UCR data repository. If the synergy between multiple representations and ensemble learning leads to top accuracy results, the latter also annihilates hopes of interpretability. Our approach FEARS opens a new way to exploit multiple representations: transformed data are stored in several linked tables following a relational schema (say relational data). The goal is to extract relevant and interpretable features from these linked tables and flatten it into a unique feature-value data table. This relational data mining process may be performed through propositionalisation.

Automatic propositionalisation - Relational data contains one root table where each row represents a statistical individual and other secondary tables containing detail records with possibly one-to-many relationships. Supervised learning from relational data (Dzeroski and Lavrac (2001)) is primarily performed through two approaches: *(i)* building full-fledged relational machine learning algorithm (Assche et al. (2006)) that directly processes relational data; *(ii)* propositionalisation (Lachiche (2017)) that flattens relational data by generating aggregate features. The single feature-value table arising from propositionalisation has the advantage to benefit from a strong arsenal of classifiers provided by decades of research literature. Since pioneering work LINUS (Lavrac et al. (1991)), many approaches for propositionalisation have been suggested: e.g., (Zelezný and Lavrac (2006); Landwehr et al. (2007)) based on frequent patterns or queries, (Kuzelka and Zelezný (2011)) uses the existential quantifier only to generate features. Other *logic-based* methods, such as RSD (Lavrac et al. (2002)) or SINUS (Lavrac and Dzeroski (1994)) tackle the propositionalisation task by constructing first-order logic features. One the other hand, the *database-inspired* methods such as POLKA (Knobbe et al. (2001)) and RELAGGS (Krogel and Wrobel (2001)) apply aggregation functions, such as *Min*, *Max*, and *Mean* to generate interpretable features. However, existing approaches proceed to propositionalisation independently to the model training and may lead to overfitting problems when complex features are generated. Our approach FEARS builds upon recent advances (Boullé et al. (2019)) on automatic propositionalisation that is able to extract complex aggregates in a regularized way.

3. Feature and representation selection with FEARS

This section describes in details the key steps of the suggested approach. Firstly, Section 3.1 presents the various transformations applied to original time series. Two possible options to recode these multiple representations into relational data are also presented. Then a propositionalisation approach is used to extract informative features from the outcoming relational data, and summarized in Section 3.2. Finally, Section 3.3 presents the algorithm for representation selection.

3.1. Multiple representations of time series in relational schema

As in (Gay et al. (2013)), we use the original time representation and we pick six additional representations among the numerous ones existing in the literature.

- **Derivatives:** We use derivatives (D) and double derivatives (DD) of the original time series. These transformations allow us to represent the local evolution of the series *(i.e., increasing / decreasing, acceleration / deceleration).*

- **Cumulative sums:** We also use simple (CUMSUM) and double (DCUMSUM) cumulative integrals of the series, computed using the trapeze method. These transformations allow us to represent the global cumulated evolution of the series.

- **Auto-correlation:** The (ACF) transformation describes the correlation between values of the signal at different times and thus allows us to represent auto-correlation structures like repeating patterns in the time series. The transformation by auto-correlation is: $\tau_{i\rho} =< (t_1, \rho_1), ..., (t_m, \rho_m) >$ where, $\rho_k = \frac{\sum_{j=1}^{j=m-k}(x_j - \bar{x}).(x_{j+k} - \bar{x})}{m.s^2}$ and where $\bar{x}$ and s^2 are the mean and variance of the original series.

- **Power Spectrum:** A time series can be decomposed in a linear combination of sines and cosines with various amplitudes and frequencies. This decomposition is known as the Fourier transform. The Power Spectrum (PS) is: $PS(\tau_i) =< (f_1, a_1), ..., (f_n, a_n) >$, where f_k represents the frequency domain, a_k the power of the signal and n the number of considered frequency values[2]. This transformation is commonly used in signal processing and encodes the original series into the frequency domain.

2. In our experiments, we used $n = m$ as default value.

The choice of representations is made according to two criteria: they have been used in the TSC literature and their computational complexity is sub-quadratic. The fast Fourier transform allows to produce ACF and PS representation in $O(m \log m)$, where m stands for the series' length.

Relational schema - We, then, gather the previously computed representations within a relational data set. The "schema" of a relational data set defines the structure of the included tables, their types *(i.e. root or secondary tables)* and their links. A large variety of possible schemes exist, and the choice of a particular schema may have a significant impact on quality of the learned classifier. As shown in Figure 2, we consider two kinds of schemas in order to encode the multiple representations of time series. The **all-in-one** schema *(Figure 2(a))* is composed by : i) a root table used to index the time series objects by using their identifiers; ii) a secondary table by domain *(time and frequency)*. This first schema gathers the representations that share a similar index in the secondary tables. The **one-each** schema *(Figure 2(b))* is composed by : i) the same root table that indexes the time series identifiers; ii) one secondary table per representation. The secondary tables contain a secondary index *(timestamp or frequency)* and store the values of each data points. The second schema stores each representation in dedicated secondary tables. These two kinds of schemas correspond to different ways of generating aggregate features: *(i)* the *"all-in-one"* schema promotes the generation of cross-representation features *(i.e. jointly based on several representations)*; *(ii)* the *"one-each"* schema prioritizes the mining of complex aggregates independently on each representation. During the learning phase, the best schema is dynamically chosen for each data set. This point is detailed in Section 3.3, where the representation selection algorithm is presented.

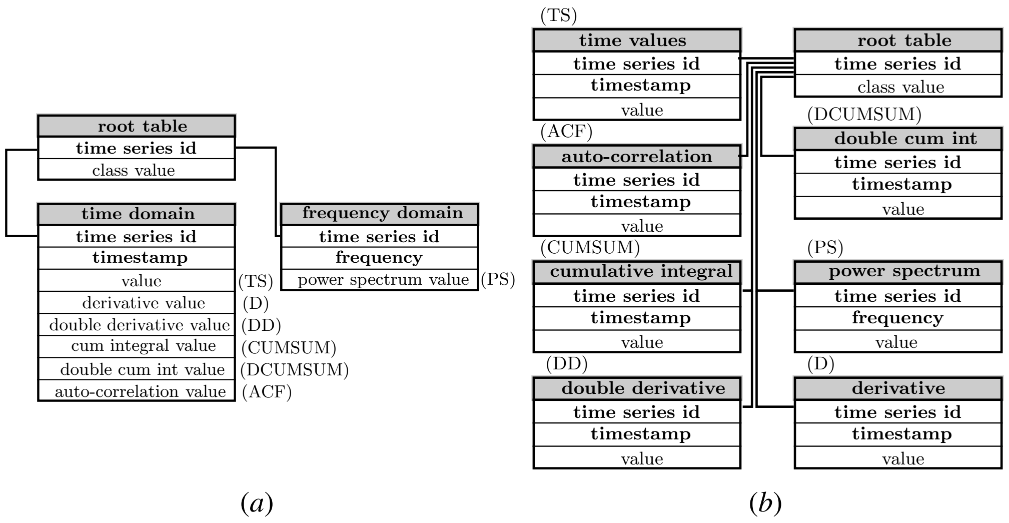


Figure 2: Relational schemas for encoding time series as relational data: (a) *all-in-one* schema and (b) *one-each* schema.

3.2. Propositionalisation, supervised feature selection and classification

Propositionalisation is the process of adding columns (features) containing information extracted from the secondary tables to the root table (Lachiche (2017)), say flattening. In the case of TSC, propositionalisation may generate different aggregate features from various representations. The *standard deviation of the derived time series taken over a given period* is an example of such aggregate feature. It is unrealistic to consider the automatic generation of all of them, therefore propositionalisation techniques exploit a restricted language for feature generation, i.e., using a finite set of construction rules. In our approach, a construction rule is similar to a function in a programming language. It is defined by its name, the list of its operands and its return value. The operands and the return value are typed. The operands can be a column of a table, the output of another rule *(i.e. another generated feature)*, or a constant coming from the training set.

Since the variables that defined time series are numerical, we use a combination of *(i)* historical aggregate functions from relational data base domain dedicated to numerical variables, namely, *min, max, sum, count (distinct), median, mean, stdev* and *(ii)* a *Selection* function to allow restriction to intervals of timestamp/frequency and values dimensions in secondary tables. Thus, our previous illustrative example (see Figure 1) can be represented by the following aggregate feature: *StdDev(Selection(derivative, 2012-04-06 01:40 < timestamp < 2015-04-06 23:20),Value)*

In this case, the *StdDev* function is applied on the output of another construction rule, namely the *Selection* function exploited to identify a particular time period. The search space for the features that can be generated consists of all possible function compositions, only limited by the type of operands of each function. Thus, the number of function compositions is not limited and the search space is infinite. In this framework, there are two important challenges to overcome: i) the combinatorial explosion for the exploration of the search space; ii) the risk of over-fitting due to the generation of arbitrarily complex features.

The FEARS approach instantiates the MODL framework (Boullé (2006, 2007); Boullé (2014); Boullé et al. (2019)) for TSC to solve these problems by introducing an evaluation criterion based on a Bayesian formalism to penalize complex sampled features. This criterion allows one to filter uninformative features by taking into account the balance between the complexity of the aggregate and its informativeness. The rest of this sub-section describes the following steps: i) the sampling of the aggregate features; ii) the filtering of the uninformative features; iii) learning a classifier. While interested readers may find full details on the Bayesian methods at use in (Boullé et al. (2019)), we report here the main underlying principles in order to produce a self-contained paper.

3.2.1. SAMPLING GENERATED FEATURES -

The optimal way for an end-user is to work with the least parameter setting. We suggest to use a single parameter K, the number of aggregate features to be sampled from the input relational data. The infinite search space can be represented by a tree structure *(see Figure 3)*. Each branch of this tree corresponds to an aggregate feature that can be drawn. The sampling of the K features is done by building this tree through sequential steps *(denoted by* 1, 2, 3, 4 *in Figure 3)*.

This very simple example corresponds to the *all-in-one* schema and represents only 3 out of 6 representations. The first step consists in choosing either a native feature belonging to the root table, or the generation of an additional feature. Here, we assume that the root table contains two native features that describe the length and the duration of the time series. An additional choice to generate an additional feature is represented by the node named *"Choice of rule"*. The second step consists in choosing the construction rule, here, the *Min* or *Max* functions. The other steps correspond to the operand choice of these two functions. Step 3 consists in choosing the secondary table and step 4 corresponds to the choice of the column on which to apply the current function. Once again, there is an additional choice that allows the algorithm to generate the input of the current function *(Min or Max)* by applying another construction rule.

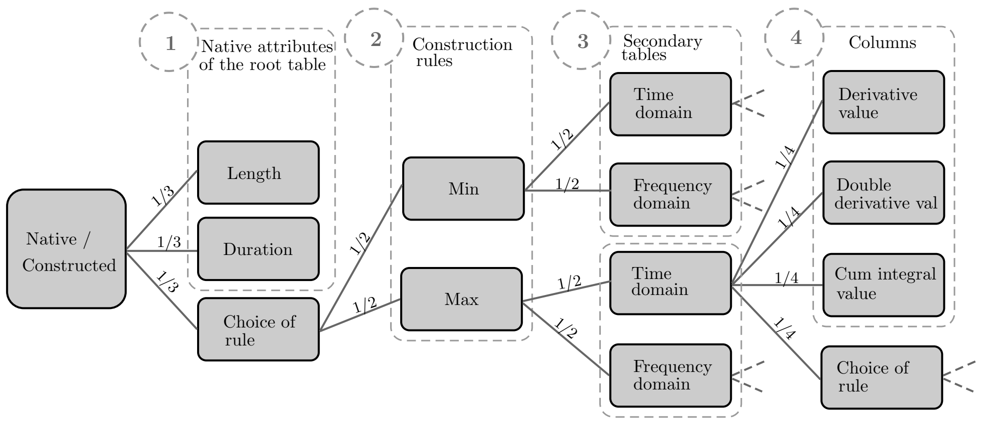


Figure 3: Feature construction tree example

The width of this tree is finite because the set of the construction rules, the secondary tables are finite. By contrast, the depth of this tree is infinite due to the potentially infinite function compositions. In order to sample K features from such a tree, the drawing of features follows a particular prior distribution which is a Hierarchy of Multinomial Distributions with potentially Infinite Depth *(HMDID)* (Boullé et al. (2019)). This HMDID distribution is represented in Figure 3 by the probabilities assigned to the edges between the nodes of the tree. The algorithmic solution consists in iteratively moving a collection of tokens down the tree, according to the HMDID distribution. The number of tokens that move forward is finite, which means that the tree is only partially and progressively explored. Consequently, this algorithm can efficiently draw aggregate features with restrictions to the available computer memory.

3.2.2. FILTERING UNINFORMATIVE FEATURES -

Once the aggregate features have been generated in the main table through propositionalisation, there is no guarantee that they are informative w.r.t. the class label. Since all features that could be generated are numerical, the supervised pre-processing step consists in partitioning the numerical features into intervals, i.e., univariate discretization.

In the MODL framework, supervised discretization of a variable X is turned into a model selection problem and solved in a Bayesian way through optimization algorithms (Boullé (2006)). According to the Maximum A Posteriori (MAP) approach, the best discretization model M_X is the one that maximizes the probability of a discretization model given the input data D_X and the target output data D_Y, i.e., $P(M_X \mid D_X, D_Y) \propto P(M_X) \times P(D_Y \mid M_X, D_X)$. The prior $P(M_X)$ and the conditional probability $P(D_Y \mid M_X, D_X)$ called the likelihood are both computed with the parameters of a specific discretization which is uniquely identified by the number of intervals, the bound of the intervals and the class frequencies in each interval. Therefore, the prior exploits the hierarchy of parameters and is uniform at each stage of the hierarchy.

The evaluation criterion used for optimization is denoted by $c(M_X)$ is made of the negative logarithm of the posterior probability (Equation 1). The prior part of the optimization criterion favors simple models with few intervals, and the likelihood part favors models that fit the data regardless of their complexity. In terms of information theory, this criterion is interpreted as coding lengths. In Equation 2, the term $L(M_X)$ represents the number of bits used to describe the model and $L(D_Y|M_X, D_X)$ represents the number of bits used to encode the target variable with the model, given the input data D_X.

$$c(M_X) = -\log(P(M_X)) - \log(P(D_Y \mid M_X, D_X)) \quad (1)$$

$$c(M_X) = L(M_X) + L(D_Y|M_X, D_X) \quad (2)$$

Based on a greedy bottom-up heuristic, a specifically designed optimization algorithm is employed in order to find the most probable model given the input data in $O(N \log N)$ time complexity, where N is the number of time series. The previously described propositionalisation algorithm draws aggregate features which are evaluated by using a specifically designed MODL criterion. Compared to the supervised preprocessing criterion $c(M_X)$, the propositionalisation criterion $c^*(M_X)$ adds a construction cost within the prior part *(see Equation 3)*. Intuitively, the construction cost $L(X)$ is even more important when the considered aggregate feature X is complex. The added construction cost modifies the balance between the prior and likelihood terms by taking into count the complexity of the evaluated feature. The construction cost $L(X)$ is recursively defined due to the multiple function compositions. In Equation 4, the term $log(K+1)$ describes the cost of choosing to generate a new feature in addition to the K native features of the root table. The term $log(R)$ describes the cost of choosing a particular construction rule among R possible rules. The recursive side appears with the term $\sum_{o \in \mathcal{R}} L(X_o)$ that describes the cost of constructing a new feature for each operand of the current construction rule $\mathcal{R}$. A natural trade-off appears: the more complex the feature is, the more it is penalized by the prior, and the higher the likelihood have to be compensated $L(X)$. Compression gain (CG) evaluates the MAP model M_X^* by comparing its coding length with the one of $M_X^{\emptyset}$ that includes a single interval *(Equation 5)*. Features with a negative CG are considered as uninformative.

$$c^*(M_X) = L(X) + L(M_X) + L(D_Y|M_X, D_X) \quad (3)$$

$$\text{where } L(X) = log(K+1) + log(R) + \sum_{o \in \mathcal{R}} L(X_o) \quad (4)$$

$$CG = 1 - \frac{c^*(M_X^*)}{c^*(M_X^{\emptyset})} \quad (5)$$

$$CG^* = 1 - \frac{\sum_{i=0}^{N} log(\hat{p}(y_i|\tau_i))}{\sum_{i=0}^{N} log(p(y_i))} \quad (6)$$

3.2.3. LEARNING A CLASSIFIER -

The MODL criterion c^* is used to pre-process the informative aggregate features by training discretisation models. Then, all these univariate preprocessing models are gathered together and used to learn a Selective Naïve Bayes (Boullé (2007)) (SNB). The SNB classifier aims to select the most informative subset of features by using a specifically designed MODL criterion. Compression gain is also defined for evaluating a classifier (CG^*) and it can be interpreted as a normalized log likelihood. Equation 6 defines CG^*, where $\hat{p}(y_i|\tau_i)$ is the probability estimated by the classifier that the time series τ_i belongs to its true class y_i, and $p(y_i)$ is the probability of the class value y_i that is empirically estimated using the entire dataset. As previously, the maximum value of CG^* is 1 which corresponds to a perfect classifier, and the values of CG^* may decrease below 0 for classifiers that are less performant than predicting the empirical probabilities of the class values.

3.3. Representation selection

The previous sections presented the first steps of our approach that aim to: i) transform the original time series into multiple representations; ii) recode these multiple representations into relational data; iii) extract informative features from the outcoming relational data and learn a classifier. All these steps are repeated several times in order to select the most informative representations and the best relational schema. Namely, the algorithm of Figure 4 is used to select simultaneously the best subset of representations. This forward backward selection algorithm is repeated two times in order to select the best relational schema (see Section 3.1). The schema for which the resulting classifier has the best compression gain is retained. Step (A) of Figure 4 consists in choosing the best starting point for the representation selection based on compression gain of the learned classifier. Step (B) of Figure 4 selects the most informative representations by adding or removing it according to a shuffled order until the compression gain is no more improved by at least ϵ.

Figure 4: Representation selection algorithm

```
Require: $X$ a set of time series, $y$ the associated labels, $\mathcal{R}ep$ the set of all possible representations of time series, $S$ a set of possible starting point for the representation selection ($\forall s \in S, s \subset \mathcal{R}ep$), $\varphi$ one of the two possible relational schema (see Section 3.1), $K$ the number of generated features from relational data, $\epsilon$ the minimal compression gain improvement to add or remove a representation.
 1: $s^* \leftarrow \emptyset$                    /*Selected representations*/
 2: $CG^* \leftarrow 0$                    /*Best compression gain*/
 3: /* (step A) Choice of the best representation subset.*/
 4: for all $s \in S$ do
 5:     $classifier \leftarrow$ Learn($X, y, s, \varphi, K$)
 6:     if CompressionGain($classifier$) $> CG^*$ then
 7:         $s^* \leftarrow s$
 8:         $CG^* \leftarrow$ CompressionGain($classifier$)
 9:     end if
10: end for
11: /* (step B) Forward backward selection algorithm */
12: while $CG^*$ is $\epsilon - improved$ do
13:     /* (step B.1) Forward step*/
14:     ForwardRepSelection($Rep, CG^*, s^*$)
15:     /* (step B.2) Backward step*/
16:     BackwardRepSelection($Rep, CG^*, s^*$)
17: end while
18: return $s^*$
```

4. Empirical validation

In this section, we present the empirical evaluation of our FEARS approach for time series classification. The experiments are performed to discuss the following questions:

$\mathcal{Q}_1$ Concerning FEARS, is there a best scheme out of all-each and all-in-one schemes? How many representations selected? Which ones? Is there a benefit in predictive performance using the representation selection procedure? And what about the time-efficiency of the whole process?

$\mathcal{Q}_2$ Are the performance of FEARS comparable with state-of-the-art TSC methods? In which conditions?

4.1. Experimental protocol

Data sets. The data sets used only contain univariate time series. Even if the FEARS approach can easily be adapted to the multi-variate case, our objective is to provide results comparable with previous work in the literature. For our experiments, we use 85 TSC data sets of the UEA/UCR repository (Bagnall et al. (2018)) that are commonly used in the literature. The repository exhibits a large panel of TSC application domains: image outline, motion sensors, simulated, etc. These data sets are provided with a pre-defined split between training and test sets, and we use it as provided. This choice is questionable for many reasons, as discussed in (Gay and Lemaire (2018)), however it is sufficient for the current proof of concept. Additional experiments as well as source code which allow full reproducibility are available from the companion web page: https://github.com/alexisbondu/FEARS.

FEARS setting - Algorithm of Figure 4 involves several parameters. In our experiments we use $S = \{[TS, ACF], [TS, CUMSUM], [TS, D], [TS, CUMSUM, DCUMSUM], [TS, D, DD], [TS, D, DD, CUMSUM, DCUMSUM], [TS, D, DD, CUMSUM, DCUMSUM, ACF]\}$. The minimal compression gain improvement to add or remove a representation is fixed to $\epsilon = 0.01$. And the MODL propositionalisation approach generates at most $K = 30000$ features.

Contenders - The proposed approach is compared to the most popular state-of-the-art approaches: **1-NN-DTW** approach consists in applying a simple one-nearest neighbor classifier using the Dynamic Time Warping (Sakoe and Chiba (1978)). The **1-NN-DTW-CV** approach is a variant that tunes the warping window size by using a cross validation. **COTE**, the Collective of Transformation-Based Ensembles (Bagnall et al. (2015)) and its successor HIVE-COTE, (Lines et al. (2018)); **BOSS**, Bag-of-SFA-Symbols (Schäfer (2015)) approach; **WEASEL**, Word ExtrAction for time SEries cLassification (Schäfer and Leser (2017)), The Shapelet Transformation **ST** approach (Hills et al. (2014)); **EE**, the Elastic Ensemble (Lines and Bagnall (2015)) approach is an ensemble classifier that combines eleven k-NN classifiers using different similarity measures *(DTW, Euclidean distance etc.)*, using different representation of time series *(time domain, and the first order derivatives etc.)*. **TSBF**, the Time Series Bag of Features (Baydogan et al. (2013)). **LS**, the Learn Shapelet (Grabocka et al. (2014)). And also neural networks approaches, **RESNET & FCN** : H. I. Fawaz et al. (Fawaz et al. (2019)) performed an extensive benchmark of the main Deep Learning architectures using the same 85 datasets. We retain the two best performing Deep Learning architectures as competing approaches in our benchmark: the **RESNET** (Wang et al. (2017)) and **FCN** (Wang et al. (2017)) architectures.

4.2. Results

Schema effect - Our approach uses two different relational schemes to transform the multiple representations of times series into relational data (“all-in-one” or “one-each” schemes). The first question our experiments answer is the effect of choosing a particular schema. The Wilcoxon signed-rank test was applied on the 85 data sets, and indicates that the “all-in-one” schema performs significantly better than the “one-each” schema ($z = -3.47$). This underlines the relevance of generation "cross-representation" aggregate features (see Section 3.1). As described above, the proposed approach is able to choose the best relational schema, based on the compression gain of the learned classifier. No significant difference can be observed by the Wilcoxon signed-rank test when comparing the systematic use of the all-in-one schema and the selection of the best schema based on compression gain. However the dynamic choice of the best schema is retained in our approach, because it results in a better ranking than the single all-in-one schema when compared to competing methods. The all-in-one schema remains the most widely used in our experiments, with 72 selections out of 85 data sets.

Representation selection - The function $Learn(.)$ of Figure 4 triggers the extraction of aggregate features from relational data and the learning of the SNB classifier. This function is called for each move (i.e. adding or a deleting a representation) in the while loop of the selection algorithm. The computing time of our approach increases linearly with the number of loops in the selection algorithm. In our benchmark, the most informative subset of representations is found by using only two loops for 71 data sets, and three loops for the remaining data sets. The number of selected representations varies depending on the data sets. Figure 5:a shows the distribution of the number of selected representations on all data sets. Surprisingly, this distribution is not uniform but has a bimodal shape. Either, the learning task requires few representations (2 or 3) or many (6 or 7). Intermediate cases are very infrequent. Similarly, the representations themselves are selected in a non-uniform way. Figure 5:b shows the distribution of the number of selection occurrences[3] for each representation. In our benchmark, the most frequently used representations are the time domain (TS) and the derivatives (D and DD). Cumulative integrals (CUMSUM, DCUMSUM) and the auto-correlation function (ACF) are less frequently used. At last, the power spectrum (PS) is the less frequently used representation. The selection algorithm aims at improving the accuracy of the learned classifier by finding the most informative subset of representations. In order to evaluate this, we measure the improvement of the classifier when it is fed (or not) by all the selected representations. Namely, we build a competing approach by selecting the best single representation based on the compression gain of the learned classifier. Figure 5:c compares, for each data set, the CG obtained selecting the best single representation against the one obtained with the entire subset of selected representations. The CG is widely improved in most cases, which underlines the efficiency of the used selection algorithm. This result is also reflected in the test accuracy of the classifiers, for which the Wilcoxon test finds a very significant advantage in the use of all selected representations ($z = -4.23$). In our experiments, FEARS generates at most $K = 30000$ informative features. In practice, this number may be lower if there are not enough distinct informative features to be generated. Figure 5:d shows the distribution of the number of generated features on all data sets. In most cases less than 2500 features are generated, and this distribution appears almost uniform for higher values. In the end, FEARS extracts interpretable features, while limiting the number of generated features in practice (K is a maximum value).

Accuracy results comparison - The presented experiments evaluate and compare the performances of FEARS and the twelve competing approaches over the 85 datasets. The Nemenyi test is used to rank and group the different time series classification approaches. This statistical test includes two successive steps. First, the Friedman test is applied to the accuracy of competing classifiers to determine whether their overall performance is similar. If this is not the case, the post-hoc test is applied to determine groups of models whose overall performance is not statistically significantly different from each other, but is significantly different from those of other groups. Figure 6:a shows the Nemenyi test applied to the 85 datasets. FEARS belongs to the fourth group and is ranked 9th. The accuracy of the FEARS approach is significantly better than the 1-NN DTW baseline. This mitigated result is not surprising: very low CG^* values obtained by FEARS on several specific data sets during the training phase are the forerunners. A tight look at which data sets we fail highlight the amount of available training series. As spotted in (Gay and Lemaire (2018)), the UEA/UCR archive contains a significant number of data sets with some singularities: very few training instances or very few training class representatives or both. In these contexts, supervised discretisation of aggregate features based on empirically observed per-class frequencies are ineffective. The lack of training instances is particularly harmful since FEARS is a regularized approach: it favors simple models when the size of the training data set is insufficient. In contrast, complex models are selected when the size of the training data set increases, allowing the regularized approach to accurately describe the data. At the end, FEARS is a conservative approach with small datasets,

3. the number of times a representation is selected divided by the total number of selected representation for all datasets

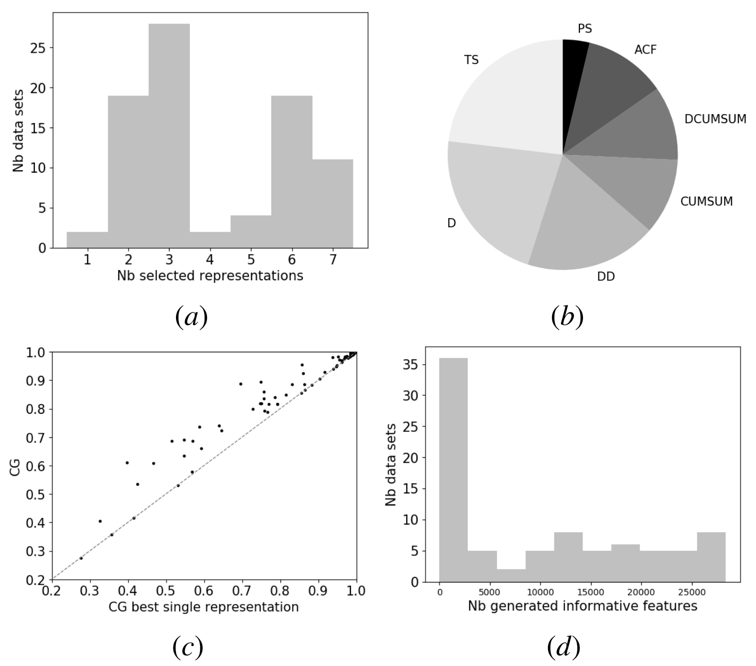


Figure 5: (a) Distribution of the number of selected representations; (b) Repartition of the representation; (c) CG obtained with the best single representation *vs.* CG of FEARS; (d) distribution of generated features.

even if it means sacrificing accuracy. This phenomenon can be observed in our benchmark. The more the data sets size increases, the more the performance of FEARS increases compared to competing approaches. Figure 6:b shows the Nemenyi test applied to the datasets that contain more than 500 training examples. In this case, the Nemenyi test is applied to the 23 biggest datasets. FEARS reaches the second rank and provides better performance than WEASEL and ResNet approaches. And HIVE-COTE remains the best approach in term of accuracy.

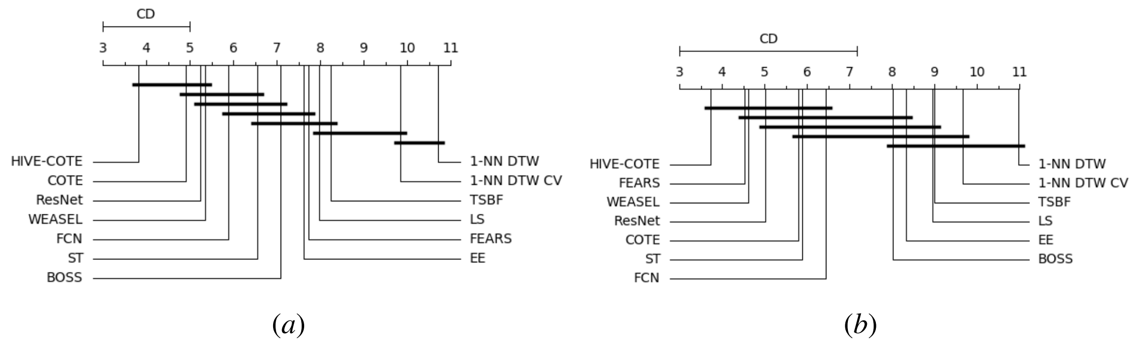


Figure 6: Nemenyi test: (a) applied to the 85 datasets; (b) applied to the data sets with $N > 500$.

Figure 7:a illustrates this by plotting the rank achieved by FEARS in the Nemenyi test against the minimum size of the training set. We can see the FEARS's rank improves monotonously when the minimum data size increases from 0 to 600 training examples. For larger sizes, there are not enough remaining datasets to make the FEARS's ranks significant. However, FEARS achieve the 4th position between 1000 and 1800 training examples, and reaches the first position when evaluated

on the 3 biggest datasets ($N > 1800$). Figures 7:a-b show the pairwise comparisons between approaches based on the Wilcoxon signed-rank test. This type of visualization allows us to compare competing approaches more precisely than the Nemenyi test. The small black squares identify pairs of approaches that do not differ significantly in performance. For instance, HIVE-COTE is significantly better than all the other competing approaches, except RestNet that reaches the 3th position in the Nemenyi test (see Figure 7:b). When evaluated on the datasets that contain at least 500 training examples, FEARS is significantly better than BOSS, EE, LS, TSBF, 1-NN DTW (CV) approaches (see Figure 7:c).

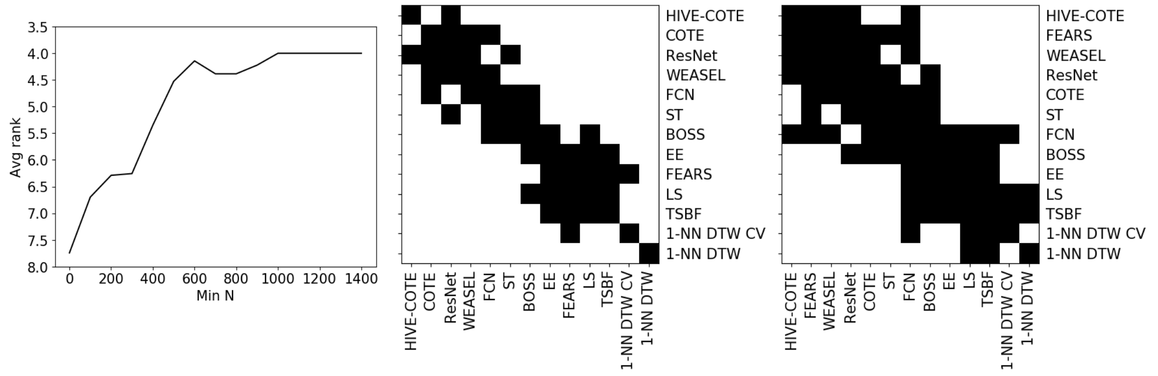


Figure 7: (a) Average ranks of FEARS in the Nemenyi test *vs.* minimal N values. And (b, c) pairwise comparison using the Wilcoxon signed-rank test: (b) all datasets; (c) $N > 500$ (competing approaches are ordered on the axes as in Figures 6:a and 6:b).

These experimental results are very encouraging and demonstrate the value of the proposed approach in term of model's accuracy. In addition, the relatively good performance of the FEARS approach must be consider regarding its computing cost.

Efficiency: running time results - In practice, the proposed approach is proving to be efficient. The overall time complexity of FEARS is in $\mathcal{O}(|\mathcal{R}ep|.K.Nlog(K.N))$, with $N, K, |\mathcal{R}ep|$ the number of training examples, generated features and representations resp. In addition, the used code source is based on a well optimized ML library that implements the MODL approaches [4]. Our experiments required limited hardware resources and were carried out in approximately 90 hours. To do this, we used a simple workstation equipped with a 2Ghz Xeon E5-2650 processor and 32GB of RAM. Figure 8:a shows the distribution of the computing times when datasets are processed on a single core. Most of the datasets are processed in less than 5 hours. And the three biggest datasets have a the most important computing times *(ElectricDevices - 49h, NonInvasiveFatalECGThorax1 - 45h, NonInvasiveFatalECGThorax2 - 43h)*. Figure 8:b plots the computing time of datasets according to their size N. This figure shows a linear trend (in log scale) of the computing time *vs.* N, that is consistent with the time complexity of the FEARS approach. At the end, FEARS provides a good compromise between the accuracy of the models and its computation time. For comparison purposes, the benchmark on deep learning approaches (Fawaz et al. (2019)) used 60 GPUs for about one month, while our experiments were performed in 90 hours by using a single 8-core CPU.

4. available at http://www.khiops.com

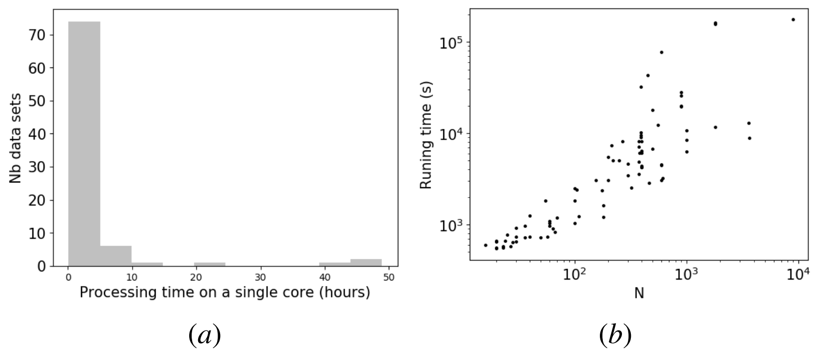


Figure 8: (a) Distribution of the running time (hours scale); (b) running time for each data set (in second) according to N the size of the training set plotted in log scale.

5. Conclusion

At the crossroads of time series classification and relational data mining communities, our contribution, FEARS is a new method that exploits multiple representations of time series for effective and efficient classification. The originality is to bring a different view on TSC – a relational view. Storing multiple representations of time series in relational data schema, interpretable feature construction/selection, and representation selection are the key concepts of FEARS. The whole process achieves very competitive accuracy results compared with recent state-of-the-art contenders on UCR/UEA benchmark data sets, provided there are enough training series. In addition, the suggested approach allows interpretable features to be extracted from the selected representations, resulting in a very advantageous compromise between (i) computation time, (ii) accuracy results and (iii) features interpretability. Thus, the suggested approach can be easily used in an industrial context, due to its high level of automation, performance, and ease of use.

The relational view of TSC comes up with a natural perspective for future work: multivariate TSC. Storing the various dimensions of multivariate series (or several alternative representations of these dimensions) in secondary tables and multivariate TSC is absorbable by FEARS. Another possible improvement of FEARS would be to enrich the relational schema with additional representations. FEARS may be combined with other machine learning approaches able to extract relevant features from the raw time series. For instance, an auto-encoder may be used to provide an additional secondary table used in our approach (but this may potentially lead to overfitting problems).

The experiments on representation selection have confirmed the prevalence of representation choice in various application domains of TSC. Since, a priori, no representation stands out from the crowd, it is up to the application domain experts to preset a starting set of potentially relevant representations. Also, the set of construction rule functions ($\min, \max$, etc.) might be fed by additional dedicated functions. Fortunately, FEARS is abstract enough to allow domain-driven customization.

References

Anneleen Van Assche, Celine Vens, Hendrik Blockeel, and Saso Dzeroski. First order random forests: Learning relational classifiers with complex aggregates. *Machine Learning*, 64(1-3):149–182, 2006.

A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with cote: The collective of transformation-based ensembles. *IEEE Trans. on Knowledge & Data Eng.*, 27(9):2522–2535, 2015.

Anthony J. Bagnall, Luke M. Davis, Jon Hills, and Jason Lines. Transformation based ensembles for time series classification. In *Proceedings of the Twelfth SIAM International Conference on Data Mining, (SDM'12), Anaheim, California, USA, April 26-28, 2012.*, pages 307–318, 2012.

Anthony J. Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn J. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining & Knowledge Discovery*, 31(3):606–660, 2017.

Anthony J. Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn J. Keogh. The UEA multivariate time series classification archive, 2018. *CoRR*, abs/1811.00075, 2018. URL http://www.timeseriesclassification.com.

Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. A bag-of-features framework to classify time series. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 35(11):2796–2802, 2013.

M. Boullé. MODL: a Bayes optimal discretization method for continuous attributes. *Machine Learning*, 65 (1):131–165, 2006.

M. Boullé. Compression-based averaging of selective naive Bayes classifiers. *Journal of Machine Learning Research*, 8:1659–1685, 2007.

Marc Boullé. Towards automatic feature construction for supervised classification. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I*, pages 181–196, 2014.

Marc Boullé, Clément Charnay, and Nicolas Lachiche. A scalable robust and automatic propositionalization approach for bayesian classification of large mixed numerical and categorical data. *Machine Learning*, 108(2):229–266, 2019.

Saso Dzeroski and Nada Lavrac. *Relational Data Mining*. Springer-Verlag, 2001.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining & Knowl. Disc.*, 33(4):917–963, 2019.

Dominique Gay and Vincent Lemaire. Should we reload time series classification performance evaluation? (a position paper). In *3rd ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (AALTD'18)*, 2018. https://arxiv.org/abs/1903.03300.

Dominique Gay, Romain Guigourès, Marc Boullé, and Fabrice Clérot. Feature extraction over multiple representations for time series classification. In *New Frontiers in Mining Complex Patterns - NFMCP@ECML-PKDD 2013, Prague, Czech Republic, September 27, 2013, Revised Selected Papers*, pages 18–34, 2013.

Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 392–401, 2014.

Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining & Knowledge Discovery*, 28(4):851–881, 2014.

Arno J. Knobbe, Marc de Haas, and Arno Siebes. Propositionalisation and aggregates. In *Principles of Data Mining and Knowledge Discovery, 5th European Conference, PKDD 2001, Freiburg, Germany, September 3-5, 2001, Proceedings*, pages 277–288, 2001.

Mark-A. Krogel and Stefan Wrobel. Transformation-based learning using multirelational aggregation. In *Inductive Logic Programming, 11th International Conference, ILP 2001, Strasbourg, France, September 9-11, 2001, Proceedings*, pages 142–155, 2001.

Ondrej Kuzelka and Filip Zelezný. Block-wise construction of tree-like relational features with monotone reducibility and redundancy. *Machine Learning*, 83(2):163–192, 2011.

Nicolas Lachiche. Propositionalization. In *Encyclopedia of Machine Learning and Data Mining*, pages 1025–1031. Springer, 2017.

Niels Landwehr, Kristian Kersting, and Luc De Raedt. Integrating naïve bayes and FOIL. *Journal of Machine Learning Research*, 8:481–507, 2007.

Nada Lavrac and Saso Dzeroski. *Inductive logic programming - techniques and applications*. Ellis Horwood series in artificial intelligence. Ellis Horwood, 1994.

Nada Lavrac, Saso Dzeroski, and Marko Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *Machine Learning - EWSL-91, European Working Session on Learning, Porto, Portugal, March 6-8, 1991, Proceedings*, pages 265–281, 1991.

Nada Lavrac, Filip Zelezný, and Peter A. Flach. RSD: relational subgroup discovery through first-order feature construction. In *Inductive Logic Programming, 12th International Conference, ILP 2002, Sydney, Australia, July 9-11, 2002. Revised Papers*, pages 149–165, 2002.

Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining & Knowledge Discovery*, 29(3):565–592, 2015.

Jason Lines, Sarah Taylor, and Anthony J. Bagnall. Time series classification with HIVE-COTE: the hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data*, 12(5):52:1–52:35, 2018.

Abdullah Mueen, Eamonn J. Keogh, and Neal E. Young. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 1154–1162, 2011.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

Patrick Schäfer. The BOSS is concerned with time series classification in the presence of noise. *Data Mining & Knowledge Discovery*, 29(6):1505–1530, 2015.

Patrick Schäfer and Ulf Leser. Fast and accurate time series classification with WEASEL. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 637–646, 2017.

Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn J. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining & Knowledge Discovery*, 26(2):275–309, 2013.

Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 1578–1585, 2017.

Filip Zelezný and Nada Lavrac. Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, 62(1-2):33–63, 2006.