Multivariate Time Series Classification: a relational way

Dominique Gay[1], Alexis Bondu[2], Vincent Lemaire[2], Marc Boullé[2], and Fabrice Clérot[2]

[1] LIM-EA2525, Université de La Réunion, France
dominique.gay@univ-reunion.fr
[2] Orange Labs, France
firstname.name@orange.com

Abstract. Multivariate Time Series Classification (MTSC) has attracted increasing research attention in the past years due to the wide range applications in e.g., action/activity recognition, EEG/ECG classification, etc. In this paper, we open a novel path to tackle with MTSC: a *relational* way. The multiple dimensions of MTS are represented in a relational data scheme, then a propositionalisation technique (based on classical aggregation/selection functions from the relational data field) is applied to build interpretable features from secondary tables to "flatten" the data. Finally, the MTS flattened data are classified using a selective Naïve Bayes classifier. Experimental validation on various benchmark data sets show the relevance of the suggested approach.

Keywords: Multivariate Time Series Classification, Feature Selection, Bayesian Modeling, Propositionalisation, Interpretable Models

1 Introduction

Multivariate Time Series Classification (MTSC) arise from many application areas [1], e.g., human activity recognition, motion/gesture classification, ECG/EEG classification, audio spectra, handwriting, manufacturing classification, etc.

For an incoming d-dimensional MTS, $\tau = \langle X^1, X^2, \ldots, X^d \rangle$ where, for $i = 1..d$, $X^i = \langle (t_{1_i}, x_{1_i}), (t_{2_i}, x_{2_i}), \ldots, (t_{m_i}, x_{m_i}) \rangle$ are univariate time series (with $x_{k_i} \in \mathbb{R}$ the value of the X^i series at time t_{k_i}), the goal is to predict the value of a categorical target variable, say label, given a training set of labeled MTS.

While the literature for *univariate* TSC is substantial [3], existing approaches generally cannot be straightforwardly translated for MTSC problems. Besides recent deep learning based approaches [13–15], various effective methods have been suggested for MTSC: e.g., considering the reputation of Dynamic Time Warping (DTW) Nearest Neighbor for the univariate case, two different generalizations for MTS have been tried [21]. With SMTS [5], Baydogan et al. proceed a two-step random forest approach for bag-of-words modeling then classification; subsequently they suggest LPS [6] which builds a bag-of-words representation based on the leaves of regression trees trained on segments extracted from

MTS; and thereafter, they also introduce AutoRegressive Forests (ARF [22]) for MTSC. Karlsson et al. [16] exploit tree ensembles over randomly selected shapelets (gRSF). Furthermore, Schäfer & Leser suggest WEASEL+MUSE [20] which extends WEASEL [19] to build discriminative features, based on symbolic Fourier approximation and bag-of-patterns, to feed a logistic regression model. They also led benchmark comparative experiments with the above representative contenders on a well-known repository of 20 MTS data sets [4]. In terms of predictive performance, the results indicates that WEASEL+MUSE and a deep learning architecture, MLSTM-FCN [15], take the lead of the benchmark even if there is no statistically significant difference of performance with gRSF, SMTS, LPS and ARF methods.

In this paper, in order to tackle with MTSC, we open and explore a new path in which multivariate time series will be seen as multi-relational data.

As a motivating example, we consider the 4-class BasicMotions MTSC data set [1]. Basic Motions (standing, walking, running and playing badminton) are the classes of the problem and are described by 6-dimensional MTS collected through 3D accelerometer data, i.e., (x, y, z), and 3D gyroscope data, i.e., $(roll, pitch, yaw)$. In this context, considering the variable $v = min(DerivativeValue(pitch))$, i.e., the minimum value the derivative transform of the 5th dimension, and its discretization into four informative intervals, the contingency table (see Table 1) indicates a perfect discrimination of the four classes. A straightforward interpretation highlights that the minimum value of the pitch speed is characteristic of the different class motions.

v	c_1	c_2	c_3	c_4
$v \leq -7.8$	**10**	0	0	0
$-7.8 < v \leq -2.2$	0	**10**	0	0
$-2.2 < v \leq -0.624$	0	0	0	**10**
$-0.624 < v$	0	0	**10**	0

Table 1. Context: 4-class BasicMotions data (40 series of length 100, over 6 dimensions – 3D from accelerometer (x, y, z) and 3D from gyroscope (roll, pitch, yaw). Class-contingency table for the discretization of the constructed variable $v = min(DerivativeValue(pitch))$, i.e., the minimum value of the derivative transform of the pitch dimension.

As far as we know, MTSC have not yet been approached from a relational data classification point of view. Our approach, called KMTS, brings a methodological contribution to MTSC literature as it generalizes the underlying concepts of the above intuitive example to efficiently extract simple and interpretable features for MTSC, as follows: *(i)*, firstly, we transform the original MTS 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.

The rest of the paper successively presents the main concepts of our KMTS approach in Section 2, the experimental validation in Section 3 and opens future perspectives after concluding in Section 4.

2 MTSC via a relational way

Our approach, KMTS, is based on *(i)* the computation of multiple yet simple representations of time series, and their storage in a relational data scheme, *(ii)* a recently suggested approach for relational data classification [10] using feature construction through propositionalisation and, supervised feature selection and classification through a selective Naïve Bayes classifier [9]. In the following, we describe these two steps with a particular attention to make the paper self-contained.

2.1 Multiple representations of MTS in relational schemes

Since [2], 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. As recent methods [18, 7] has proven to achieve top accuracy results on transformed univariate time series, we also generates six simple transformations of the dimensions of MTS commonly used in the literature in addition to the original representation:

- **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 (S) and double (SS) cumulative Sums 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:

$$\tau_{i\rho} =< (t_1, \rho_1), ..., (t_m, \rho_m) > \quad \text{where} \quad \rho_k = \frac{\sum_{j=1}^{j=m-k}(x_j - \bar{x}).(x_{j+k} - \bar{x})}{m.s^2}$$

  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 (by default $n = m$). This transformation is commonly used in signal processing and encodes the original series into the frequency domain.

In order to keep the whole procedure time-efficient, among the numerous representations existing in the literature, we picked some of thoses that can be computed with at most sub-quadratic time complexity: e.g., the fast Fourier

transform allows to produce ACF and PS representation in $O(m \log m)$, where m stands for the series' length.

To gather the various computed representations, we investigate two different relational data schemes. In every case, the root table is made of two attributes (columns), the series ID and the class value. Depending on the scheme, secondary tables are designed as follows:

- one representation per secondary table (i.e. 7 tables). In this scheme, in Figure 1, each table is described by the following attributes: the series' ID (linked to the one from the primary table), the "x-axis" (i.e. time or frequency for PS representation), plus one column for each dimension. We refer to this scheme as 7rep_7T.
- all-in-one scheme: only one secondary table containing all representations of all dimensions of the MTS, i.e., $7 \times d$ attributes plus the series ID and the x-axis. We refer to this scheme as 7rep_1T.

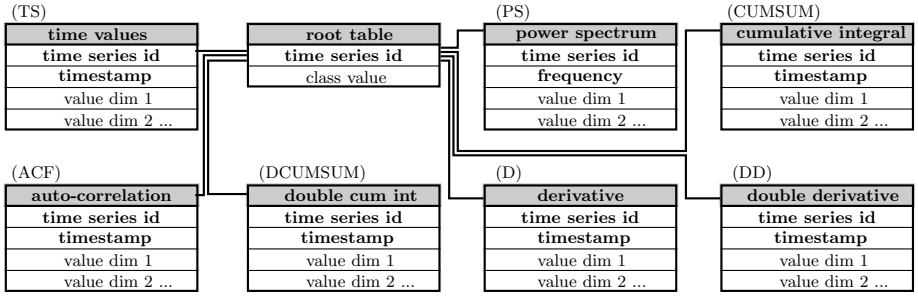

Fig. 1. Relational scheme, 7rep_7T, where each of the seven secondary tables holds a representation of the MTS.

2.2 KMTS: interpretable feature selection and classification

Feature construction through propositionalisation - In order to build features from secondary tables, we use propositionalisation; that is the process of adding columns containing information extracted from secondary tables to the root table [17]. For the MTS case, propositionalisation may generate different aggregate features from various representations of the multiple dimensions. The introductive variable v, i.e., the *minimum of the derived time series in the fifth dimension* is an example of such aggregate feature. To avoid untractable search space, propositionalisation techniques usually 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 define time series are numerical, and for interpretability purposes, we use a combination of:

- *(i)* historical and interpretable aggregate functions from relational data base domain dedicated to numerical variables, namely, *min, max, sum, count (distinct), median, mean, stdev.*
- *(ii)* a *Selection* function to allow restriction to intervals of timestamp/frequency and value variables in secondary tables.

Thus, another example of aggregate feature using the selection operator could be: *Max(Selection(derivative, 14 < timestamp < 69),ValueDim5)*, i.e., the maximum value of the derivative transform of dimension 5, in the time interval $[14; 69]$. Here, the Max function is applied on the output of another construction rule, the *Selection* exploited to identify a particular time period.

In this context, 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. Therefore, 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.

In order to avoid non-trivial parameter setting in the exploration of the search space, 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 where each branch of the 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 2).*

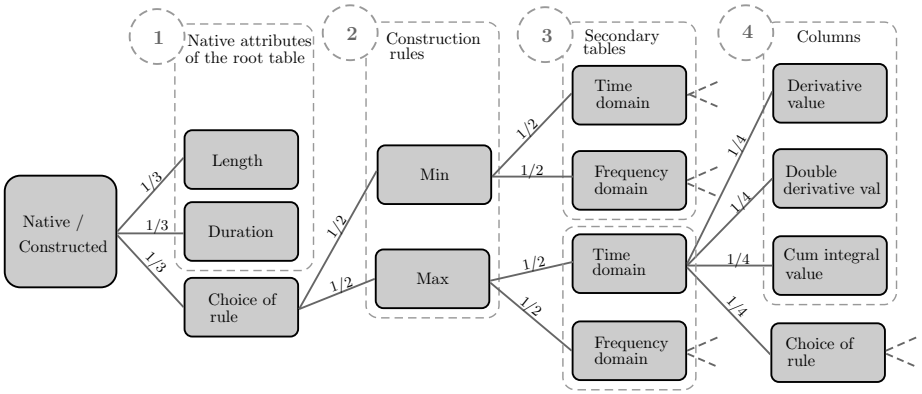

Fig. 2. Feature construction tree example for one dimension.

Due to space limitation, in Figure 2, we only consider one dimension and 3 out of 6 representations in the *all-in-one* schema. The sequential steps for feature sampling are:

- (step 1) 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"*.
- (step 2) 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
- (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.

While the width of the feature construction tree is finite because the set of the construction rules and the secondary tables are finite, by contrast, the depth of the 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)* ([10]). This HMDID distribution is represented in Figure 2 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.

Feature selection through supervised discretisation and classification - After propositionalisation, aggregate features of the main table are not guaranteed to be class-informative. As all generated features are numerical due to the nature of aggregate functions, a supervised pre-processing step is led for filtering uninformative features and partition them into intervals, that is univariate discretisation.

In the Bayesian framework [8], supervised discretisation of a variable X is seen as a model selection problem and solved in Bayesian way through optimization algorithms. According the Maximum A Posteriori (MAP) approach, the best discretisation model M_X is the one that maximizes the probability of a discretisation model given the input data D, i.e., $P(M_X \mid D) \propto P(M_X) \times P(D \mid M_X)$. The prior $P(M_X)$ and the likelihood $P(D \mid M_X)$ 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.

Switching to negative logarithm refers to information theory and defines our evaluation criterion, noted c (for cost in Eq. 1).

$$c(M_X) = -\log(P(M_X)) - \log(P(D \mid M_X)) = L(M_X) + L(D|M_X) \quad (1)$$

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: the term $L(M_X)$ represents the number of bits used to describe the model and $L(D|M_X)$ represents the number of bits used to encode the target variable with the model, given the model M_X. Greedy bottom-up algorithms [8] allows 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.

In order to avoid the construction of unnecessary complex features, we add a construction cost (related to the propositionalisation procedure) to the prior part of c, resulting in $c^*(M_X)$ (Eq. 2). 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 3, the term $log(K+1)$ describes the cost of choosing to generate a new feature in addition to the K native features (if any) 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 (Eq. 2). Features with a negative CG are considered as uninformative.

$$c^*(M_X) = L(X) + L(M_X) + L(D|M_X) \quad \text{and} \quad CG = 1 - \frac{c^*(M_X^*)}{c^*(M_X^\emptyset)} \quad (2)$$

$$\text{where } L(X) = log(K+1) + log(R) + \sum_{o \in \mathcal{R}} L(X_o) \quad (3)$$

Thus, the cost criterion c^* is used to pre-process the informative aggregate features by training discretisation models. Then, all these univariate preprocessied models are gathered together and used to learn a Selective Naïve Bayes [9] (SNB). The SNB classifier aims to select the most informative subset of features by using a specifically designed compression-based criterion. This way, the whole KMTS procedure is regularized to avoid unnecessary complex features and models, thus avoiding over-fitting.

3 Experimental validation

The experimental evaluation of our approach KMTS are performed to discuss the following questions:

Q_1 Concerning KMTS, how does the predictive performance evolve w.r.t. the number generated features and relational schemes? How many relevant features are selected? Are there preferred dimensions/representations for feature selection? And what about the time efficiency of the whole process?

Q_2 Are the performance of KMTS comparable with state-of-the-art MTSC methods?

Experimental protocol & data sets - Most of the literature are based on M. Baydogan 20 data sets [4]. Recently, in 2018, pursuing the success of the largest univariate TSC repository, the UEA team also released a 30 MTSC repository [1] with some overlapping with Baydogan repository. Both repositories exhibit a large variety of MTSC application domains with various numbers of dimensions, classes and series' lengths. Predefined train/test sets are provided and we used it per se.

3.1 Accuracy evolution w.r.t. the number of features

We study the evolution of accuracy w.r.t. K , the number of extracted features on the 30 data sets of UEA repository [1]. In Table 2, we report accuracy results of KMTS for increasing $K = 10, 100, 1000, 10000$ on original MTS, i.e., without transformations (notice that similar behaviors are observed when using the 7 representations). As expected, accuracy increases with K. While we can expect better accuracy from even more generated features (but with increasing computational time), a few more training MTS seem to also improve accuracy. This can be seen in the $10 - CV$ column (Table 2), where we report 10-folds cross-validation accuracy results. Indeed, when using 90% of available data for training, KMTS achieves better average results on 23/30 data sets (e.g., for Ering and Handwriting data). Another important observation is about the "stable-with-K" but poor accuracy results (for AtrialFibrillation, FaceDetection, FingerMovements, HandMovementDirection MotorImagery, SelfRegulationSCP2 and StandWalkJump data). For these data, KMTS found no class-informative attributes out of the 10000 generated, the major class is predicted, therefore the bad accuracies. Since KMTS is a regularized approach based on estimated per-class frequencies, data with (very) small training set size (AtrialFibrillation, StandWalkJump, ...) are a difficult task. For FaceDetection data, we may conjecture that KMTS has high bias, thus aggregate features are not the good way to tackle with. Notice that generalizations of DTW-NN also obtain poor accuracy results on these data sets [1].

Data sets	Train	Test	Dim	Length	Classes	Original representation					7 representations	
						$K = 10$	$K = 100$	$K = 1000$	$K = 10000$	$K = 10000$ (10-CV)	7rep_1T	7rep_7T
ArticularyWordRecognition	275	300	9	144	25	0.2467	0.9500	0.9800	0.9833	0.9878±0.0137	0.9800	0.9767
AtrialFibrillation	15	15	2	640	3	0.3333	0.3333	0.3333	0.3333	0.3333±0.0000	0.3333	0.3333
BasicMotions	40	40	6	100	4	0.9750	1.0000	1.0000	1.0000	1.0000±0.0000	0.9250	1.0000
CharacterTrajectories	1422	1436	3	182	20	0.7946	0.9554	0.9735	0.9735	0.9811±0.0055	0.9708	0.9861
Cricket	108	72	6	1197	12	0.7639	0.9583	0.9583	0.9861	0.9833±0.0255	0.9583	0.9722
DuckDuckGeese	60	40	1345	270	5	0.2000	0.3000	0.4800	0.4600	0.5100±0.1640	0.3800	0.4600
EigenWorms	128	131	6	17984	5	0.5954	0.7481	0.8168	0.8550	0.8725±0.0670	0.8779	0.9237
Epilepsy	137	138	3	206	4	0.7681	0.9493	0.9710	0.9710	0.9855±0.0178	0.9783	0.9783
ERing	30	30	4	65	6	0.4370	0.5481	0.7111	0.7148	0.9733±0.0200	0.7926	0.8037
EthanolConcentration	261	263	3	1751	4	0.2510	0.3840	0.4411	0.4335	0.4444±0.0596	0.2814	0.4183
FaceDetection	5890	3524	144	62	2	0.5000	0.5000	0.4997	0.4997	0.5364±0.0114	0.5119	0.5000
FingerMovements	316	100	28	50	2	0.4900	0.4900	0.4900	0.4900	0.4951±0.0060	0.4900	0.4900
HandMovementDirection	320	147	10	400	4	0.2027	0.2027	0.2027	0.2027	0.2993±0.0062	0.2027	0.2027
Handwriting	150	850	3	152	26	0.0835	0.1600	0.2506	0.2941	0.5800±0.0366	0.3129	0.3094
Heartbeat	204	205	61	405	2	0.6439	0.7073	0.7220	0.7610	0.7677±0.0609	0.7220	0.7463
InsectWingbeat	30000	20000	200	78	10	0.2454	0.5664	0.6576	0.6618	0.6679±0.0062	0.4349	0.6645
JapaneseVowels	270	370	12	29	9	0.7000	0.9595	0.9541	0.9595	0.9703±0.0191	0.7811	0.9595
Libras	180	180	2	45	15	0.4500	0.7278	0.8167	0.8167	0.9000±0.0468	0.8944	0.9222
LSST	2459	2466	6	36	14	0.4700	0.5264	0.5446	0.5568	0.5614±0.0161	0.6269	0.5714
MotorImagery	278	100	64	3000	2	0.5000	0.5000	0.5000	0.5000	0.4973±0.0054	0.5000	0.5000
NATOPS	180	180	24	51	6	0.6000	0.7833	0.7667	0.9222	0.9389±0.0324	0.8444	0.8611
PEMS-SF	267	173	963	144	7	0.7688	0.9538	0.9769	1.0000	1.0000±0.0000	0.9884	0.9884
PenDigits	7494	3498	2	8	10	0.5560	0.8473	0.8988	0.9108	0.9562±0.0055	0.9551	0.9397
PhonemeSpectra	3315	3353	11	217	39	0.0820	0.0835	0.0892	0.0957	0.1113±0.0068	0.1712	0.1807
RacketSports	151	152	6	30	4	0.5066	0.7763	0.8224	0.8816	0.9106±0.0338	0.8026	0.8224
SelfRegulationSCP1	268	293	6	896	2	0.7850	0.8225	0.8055	0.8123	0.8271±0.0486	0.8191	0.9044
SelfRegulationSCP2	200	180	7	1152	2	0.5000	0.5000	0.5000	0.5000	0.5000±0.0000	0.5000	0.5000
SpokenArabicDigits	6599	2199	13	93	10	0.5730	0.8549	0.9541	0.9700	0.9779±0.0037	0.9741	0.9718
StandWalkJump	12	15	4	2500	3	0.3333	0.3333	0.3333	0.3333	0.2333±0.1528	0.3333	0.3333
UWaveGestureLibrary	120	320	3	315	8	0.2250	0.7750	0.8813	0.9000	0.9432±0.0384	0.8969	0.8750

Table 2. Accuracy results for KMTS with K, incremental number of extracted features on original data, all dimensions in a secondary table and using 7 representations in the two suggested schemes (1 secondary table for all dimensions and their representations vs 1 secondary table for each representation).

3.2 About relational schemes

The two last columns of Table 2 reports accuracy results of KMTS for $K = 10000$ using the two suggested relational data schemes 7rep_1T and 7rep_7T. Considering Win-Tie-Loss versus original MTS data with $K = 10000$, 7rep_1T scores 11-6-13 and 7rep_7T scores 13-9-8. The scheme with seven representations stored in seven secondary tables take the advantage in terms of accuracy results and we focus on this scheme for the rest of the experiments. This also confirms the benefit of using additional simple representations of original times series to improve predictive performance.

3.3 Distribution of selected features, representations and dimensions

As KMTS select informative features to build a Naïve Bayes classifier, we study the distribution of informative and selected features for each data set in Figure 3. For some data sets, no informative attribute is found and poor accuracy results are obtained as explained earlier. For most of the other data sets, more than 100 informative features are found except for Ering, DuckDuckGeese which contains a small number of training series. Furthermore, the number selected features (that are embarked in the SNB classifier) are generally an order of magnitude lesser than the number of informative ones.

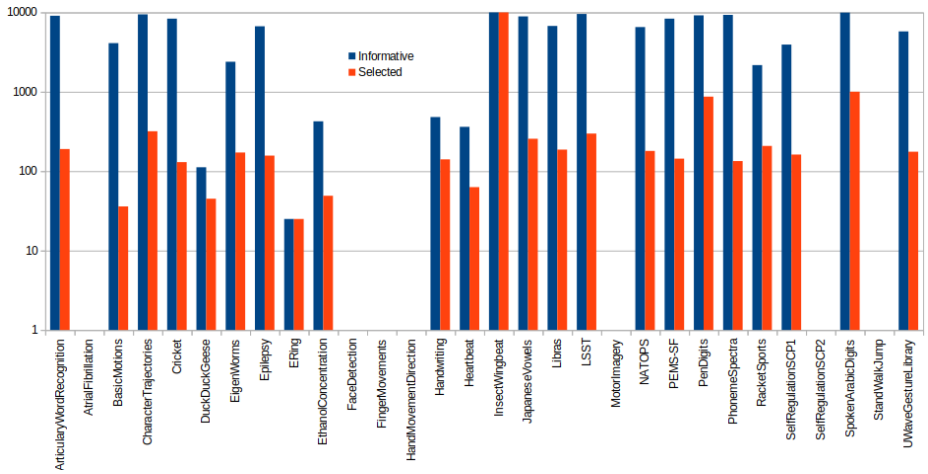

Fig. 3. Distribution of informative and selected features among the $K = 10000$ generated features, for each data set.

In Figure 4, we study the relative distribution of the selected features into the seven representations for each data set. In most cases, all the seven representations are present in the selected features. A few exceptions stand: e.g.,

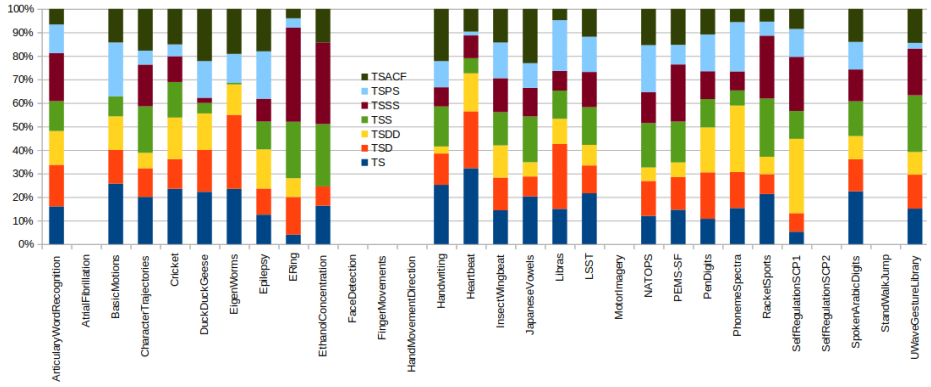

Fig. 4. Relative distribution of the seven representations among selected features in SNB for each data set.

there is no SS features for BasicMotions and EigenWorm data and relatively few PS features for HeartBeat and UWave data. Concerning the distribution of the selected features among the dimensions of MTS, we compute the percentage of dimensions not used in selected attributes: for most the data sets, all dimensions are present in the selected features, except for DuckDuckGeese (resp. Heartbeat and PEMS-SF) for which 95% (resp. 54% and 81%) of the available dimensions are unused.
These two studies also show that even if there is no "killing" representations or dimensions, the relative importance of representations and dimensions in the selected features is clearly different depending on the data set at hand and thus has to be investigated further for possible accuracy improvement; we postpone this idea for future work.

3.4 Running time

All experiments are run under Ubuntu 18.04 using an Intel Core i5-6400 CPU@ 2.70GHz x4 and 16Go RAM. The overall time complexity of KMTS comes from the relational data classification method [10] (discretisation plus feature selection through selective Naïve Bayes) and is $\mathcal{O}(K.Nlog(K.N))$, where K is the number of generated features and N the number of training MTS. In practice, with a small computational time overhead to compute the 7 representations, KMTS is efficient as shown in Figure 5. For most of the UEA data sets, KMTS (with $K = 10000$) runs in less than 100s. For InsectWingbeat, the biggest data set with 50000 MTS, about 3 hours are needed. In Figure 6, we report the evolution of running time w.r.t. K for each data sets. As we observed earlier that increasing K leads to better accuracy, it is good to notice that it also means additional computational cost. For example, for InsectWingbeat, setting $K = 10^5$ to reach better accuracy will demand about $10^5 s$ of computation.

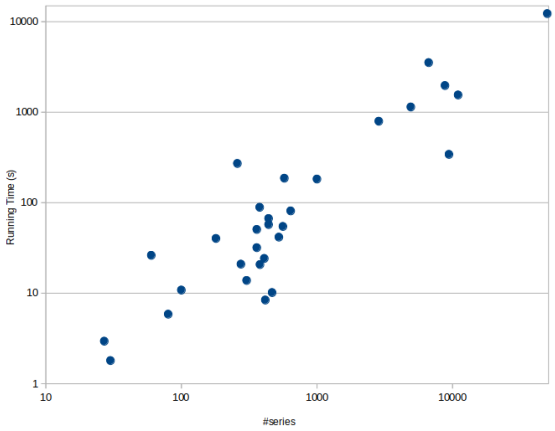

Fig. 5. Average running time results (10-cross-validation) vs. data set size (number of series) of KMTS on original representations for $K = 10000$ features on UEA repository data sets.

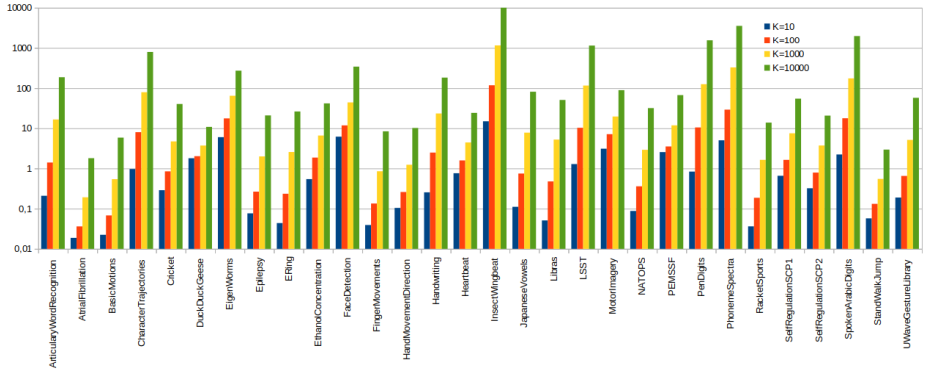

Fig. 6. Average running time results (10-CV) of KMTS on original representations for incremental K number of features on UEA repository data sets.

3.5 Predictive performance comparison with state-of-the-art

In order to compare the predictive performance of KMTS with state-of-the-art MTSC methods, we use the Baydogan repository [4] that is composed of 20 data sets (6 of which are also in the UEA repository). KMTS with $K = 10000$ using 7rep_7T is compared with 9 contenders: WEASEL [19], SMTS [5], LPS [6], ARF [22], DTWi [21], ARK [11], gRSF [16], MLSTM-FCN [15] and MUSE. All results are taken from Schäfer & Leser MUSE paper [20]. Full results are reported in Table 3 and the critical difference diagram [12] stemming from Friedman test with post-hoc Nemenyi test is shown in Figure 7.

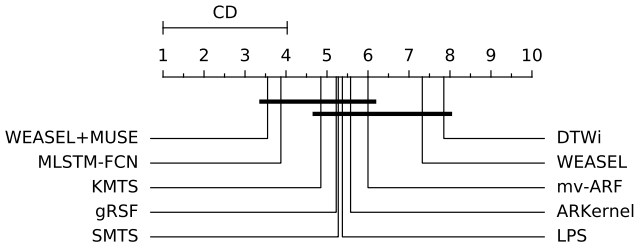

Fig. 7. Critical difference diagram

KMTS rises to the third place in terms of mean rank, just after MUSE and MLSTM-FCN. Since the critical difference diagram indicates that no significant difference of performance is found on these benchmark data sets, KMTS is among the best MTSC methods of the literature.

4 Conclusion & Perspectives

Our methodological contribution, KMTS, explores a relational way for multivariate time series classification (MTSC). Storing multiple representations of MTS in relational data scheme and interpretable feature construction/selection are the key ideas of KMTS, which end up with efficient and effective classification of MTS. The whole process achieves very competitive accuracy results compared with recent state-of-the-art contenders on benchmark data sets. In addition, the suggested approach allows interpretable features to be extracted from the dimensions of MTS and their alternative representations, resulting in a very advantageous compromise between (i) computation time, (ii) accuracy results and (iii) features interpretability.

To achieve better accuracy results, KMTS could be improved in many ways: *(i)* the ending Bayesian classifier could be swapped for e.g., ensemble methods like random forests or xgboost; *(ii)* a closer look at the data domain where KMTS

Data sets	WEASEL	SMTS	LPS	ARF	DTWi	ARK	gRSF	MLSTM-FCN	MUSE	KMTS
ArabicDigits	0.9455	0.9640	0.9710	0.9520	0.9080	0.9880	0.9750	0.9900	0.9918	0.9836
AUSLAN	0.7586	0.9470	0.7540	0.9340	0.7270	0.9180	0.9550	0.9500	0.9909	0.9607
CharTraj	0.9738	0.9920	0.9650	0.9280	0.9484	0.9000	0.9940	0.9900	0.9734	0.9949
CMUsubject16	0.9655	0.9970	1.0000	1.0000	0.9300	1.0000	1.0000	1.0000	1.0000	0.9655
DigitShapes	1.0000	1.0000	1.0000	1.0000	0.9375	1.0000	1.0000	1.0000	1.0000	1.0000
ECG	0.8500	0.8180	0.8200	0.7850	0.7900	0.8200	0.8800	0.8700	0.8800	0.8300
JapVowels	0.7892	0.9690	0.9510	0.9590	0.9622	0.9840	0.8000	1.0000	0.9757	0.9730
KickvsPunch	0.8000	0.8200	0.9000	0.9760	0.6000	0.9270	1.0000	0.9000	1.0000	0.6000
Libras	0.7280	0.9090	0.9030	0.9450	0.8880	0.9520	0.9110	0.9700	0.8944	0.9487
LP1	0.8000	0.8560	0.8620	0.8240	0.7600	0.8600	0.8400	0.8000	0.9400	0.9600
LP2	0.6330	0.7600	0.7040	0.6260	0.7000	0.6340	0.6670	0.8000	0.7333	0.5000
LP3	0.6670	0.7600	0.7200	0.7700	0.5666	0.5670	0.6330	0.7300	0.9000	0.7667
LP4	0.8670	0.8950	0.9100	0.9060	0.8667	0.9600	0.8667	0.8900	0.9600	0.8933
LP5	0.5900	0.6500	0.6900	0.6800	0.5400	0.4700	0.4500	0.6500	0.6900	0.6300
NetFlow	0.9326	0.9770	0.9680	-	0.9756	-	0.9140	0.9500	0.9382	0.9869
PenDigits	0.8338	0.9170	0.9080	0.9230	0.9270	0.9520	0.9320	0.9700	0.9128	0.8915
Shapes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
UWave	0.7627	0.9410	0.9800	0.9520	0.9158	0.9040	0.9290	0.9700	0.9159	0.9531
Wafer	0.9911	0.9650	0.9620	0.9310	0.9743	0.9680	0.9920	0.9900	0.9967	0.9900
WalkvsRun	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
mean rank	7.3250	5.2750	5.3750	6.0000	7.8500	5.5750	5.2250	3.8750	3.5500	4.8500

Table 3. Accuracy results comparison with state-of-the-art MTSC methods on Baydogan repository [4]. Results are reported from [20].

fails to find informative features, could help in finding the adequate representations and aggregate functions commonly used by domain experts in their respective domain; *(iii)* KMTS could also be wrapped in a feed forward/backward selection procedure to focus on the most informative dimensions and representations as suggested in [7].

References

1. Bagnall, A.J., Dau, H.A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Keogh, E.J.: The UEA multivariate time series classification archive, 2018. CoRR **abs/1811.00075** (2018), `http://timeseriesclassification.com`
2. Bagnall, A.J., Davis, L.M., Hills, J., Lines, J.: 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. pp. 307–318 (2012)
3. Bagnall, A.J., Lines, J., Bostrom, A., Large, J., Keogh, E.J.: 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)
4. Baydogan, M.G.: Multivariate time series classification data sets (2019), `http://www.mustafabaydogan.com`
5. Baydogan, M.G., Runger, G.C.: Learning a symbolic representation for multivariate time series classification. Data Mining & Knowledge Discovery **29**(2), 400–422 (2015)
6. Baydogan, M.G., Runger, G.C.: Time series representation and similarity based on local autopatterns. Data Mining & Knowledge Discovery **30**(2), 476–509 (2016)
7. Bondu, A., Gay, D., Lemaire, V., Boullé, M., Cervenka, E.: FEARS: a feature and representation selection approach for time series classification. In: Proceedings of

The 11th Asian Conference on Machine Learning, ACML 2019, 17-19 November 2019, Nagoya, Japan. pp. 379–394 (2019)
8. Boullé, M.: MODL: a Bayes optimal discretization method for continuous attributes. Machine Learning **65**(1), 131–165 (2006)
9. Boullé, M.: Compression-based averaging of selective naive Bayes classifiers. Journal of Machine Learning Research **8**, 1659–1685 (2007)
10. Boullé, M., Charnay, C., Lachiche, N.: A scalable robust and automatic propositionalization approach for bayesian classification of large mixed numerical and categorical data. Machine Learning **108**(2), 229–266 (2019)
11. Cuturi, M., Doucet, A.: Autoregressive kernels for time series. CoRR **abs/1101.0673** (2011), `https://arxiv.org/abs/1101.0673`
12. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. JMLR **7**, 1–30 (2006)
13. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.: Deep learning for time series classification: a review. Data Mining & Knowl. Disc. **33**(4), 917–963 (2019)
14. Hsu, E., Liu, C., Tseng, V.S.: Multivariate time series early classification with interpretability using deep learning and attention mechanism. In: Advances in Knowledge Discovery and Data Mining - 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part III. pp. 541–553 (2019)
15. Karim, F., Majumdar, S., Darabi, H., Harford, S.: Multivariate lstm-fcns for time series classification. Neural Networks **116**, 237–245 (2019)
16. Karlsson, I., Papapetrou, P., Boström, H.: Generalized random shapelet forests. Data Mining & Knowledge Discovery **30**(5), 1053–1085 (2016)
17. Lachiche, N.: Propositionalization. In: Encyclopedia of Machine Learning and Data Mining, pp. 1025–1031. Springer (2017)
18. Lines, J., Taylor, S., Bagnall, A.J.: 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)
19. Schäfer, P., Leser, U.: 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. pp. 637–646 (2017)
20. Schäfer, P., Leser, U.: Multivariate time series classification with WEASEL+MUSE. CoRR **abs/1711.11343** (2017), `http://arxiv.org/abs/1711.11343`
21. Shokoohi-Yekta, M., Wang, J., Keogh, E.J.: On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In: Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, BC, Canada, April 30 - May 2, 2015. pp. 289–297 (2015)
22. Tuncel, K.S., Baydogan, M.G.: Autoregressive forests for multivariate time series modeling. Pattern Recognition **73**, 202–215 (2018)