Multivariate time series classification based on M-histograms and multi-view learning

Angéline Plaud, Engelbert Mephu Nguifo, Jacques Charreyron

Multivariate time series classification based on M-histograms and multi-view learning *

Angéline Plaud[1,2], Engelbert Mephu Nguifo[1], and Jacques Charreyron[2]

[1] LIMOS, CNRS, University Clermont Auvergne, Clermont-Ferrand, France
[2] Michelin, Clermont-Ferrand, France
angeline.plaud@isima.fr

Abstract. Univariate time series (UTS) classification has been reported in several papers, where various efficient models have been proposed. Such models are often inadequate for multivariate time series (MTS) classification. MTS emerged with the multiplication of sensors that record large amounts of high-dimensional data, characterized by several dimensions, variable lengths, noise and correlations between dimensions. MTS classification is a challenging task, and few works have been devoted to complex data. In this paper, we propose a novel subspace model that combines M-histograms and multi-view learning together with an ensemble learning technique to handle MTS classification task. The M-histograms is a statistical tool, efficient for data visualization, that can reveal mutual information from different dimensions. Thus it can be suitable for MTS data encoding. Multi-view learning is known as data integration from multiple feature sets, as it is the case with MTS data, and multiple views also provide complementary information. The new combined model provides a MTS encoding that can outperform other time series encoding such as Symbolic Aggregate approXimation (SAX) w.r.t. to the experimental comparison that was conducted. We also benchmark our method with some state-of-art methods devoted to MTS, and discuss the results obtained and the main properties of our model.

Keywords: Multivariate time series · Classification · Mgram · ensemble learning · multi-view learning.

1 Introduction

A time series is a set of values indexed versus time representing the measures of a phenomenon. When only one sensor is used to record values, this gives rise to univariate time series (UTS). When multiple sensors record values at each time index, we obtain a multivariate time series (MTS). Different research areas like medicine [1], smart homes [3], automotive industry [2] are interested by MTS data. The main task is generally classification of MTS data where each dimension is often related to one sensor. Unlike UTS, MTS are characterized

* Supported by Michelin and ANRT.

by interactions between dimensions as by the interplay in time, such that UTS classification models are not suitable for MTS data.

WEASEL_MUSE [13] and SMTS [12] are among the recent and efficient models dedicated to MTS classification. The first one is based on Fourier transformation, symbolic representation and rolling windows. The second is a random forest transforming an MTS into a string. Those methods may suffer from scalability (time complexity) especially for long MTS, such that MTS classification is still challenging.

In this paper, we propose a novel MTS classifier by combining multi-view learning principle with a statistical model M-histograms. The M-histogram, also called Mgram, is a statistical model that allows the visualization of the distribution among data. It enables the projection of the MTS into an M-dimension array of size chosen by the user. Here, we are using it to reduce the data and to extract mutual information among the dimensions. Additionally, each dimension in an MTS is a type of variable that corresponds to one view of the data, such that MTS can be projected as a multi-view [23] of the time series data. This idea, to our knowledge, has not yet been reported in the literature.

This combination gives rise to an ensemble learning classifier by exploring different multiple views of the MTS. After the formalization of our MTS classifier model, we then experimentally demonstrate that this model is efficient compared to several MTS classifier methods, and discuss his robustness. In addition, the Mgram provides a high level data adaptive representation of MTS [5] that has shown to be competitive compared to the standard Symbolic Aggregate approXimation (SAX).

We organized this paper such that Section 2 provides a brief background on UTS and MTS. Section 3 summarizes related work. Section 4 describes the approach. Section 5 presents the results of the Mgrams models and finally Section 6 is the conclusion.

2 Background

2.1 Multivariate time series

Time series is a set of values indexed versus time, which represents the evolution of a phenomenon. The difference between a univariate time series (UTS) and a multivariate time series (MTS) is the number of values available at each time index, as shown in Fig.1a. Those multiple values at one timestamp increase the complexity of the MTS analysis as there are interactions between them. So UTS methods that are reported in the literature are not adequate for MTS without any data transformation.

An MTS X^n has M attributes or variables or dimensions at each timestamp t of the T observations, where $T \in [1, \tau]$ and X^n is the *n-th* MTS, where $n = [1, .., N]$, such that :

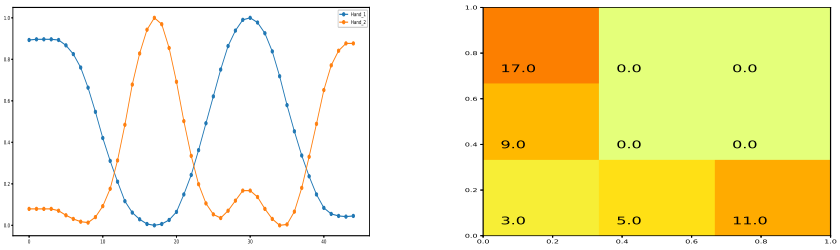
$$X_m^n = [x_m^n(1), ..., x_m^n(T)] \tag{1}$$

We can consider each of the dimensions as a univariate time series (UTS) and so, an MTS as an ensemble of UTS, which *have interactions* between them. Furthermore, when we work with MTS, we have to deal with their dimensions but also with their lengths. There are three possible cases with MTS :

1. *MTS with the same length* for all of their dimensions and for all the MTS of the dataset. This is the most common case in the literature.
2. *MTS with same length for an UTS and variable lengths between different dimensions.* All the lengths are the same inside an UTS but are variable from one dimension to another. This is the case with one sensor recording different dimensions like an accelerometer.
3. *UTS and MTS with variable lengths.* This is the case with different sensors recording data with a different time indexer and producing more or fewer data per dimension.

Our method can handle the first and the second case. The third case is outside of the scope of this work.

2.2 Mgrams



(a) Example of a 2-dimension MTS from the Libras dataset[12]

(b) Transformation of the MTS a into a bigram within 3 bins on each axis

Fig. 1: Example of transformation from a MTS to a bigram

An M-histogram, also called Mgram, is a visualization model of the density function among data. It allows the projection of the MTS into an M-dimension array. It usually takes as parameters to tune, the ranges $[min, max]$ of the value and the number of bins for each dimension.

In addition, as a global parameter, we also have the M of Mgrams. When $M = 1$, we have a 1-gram, also known as a histogram. When $M = 2$, we called it a bigram. As Mgrams is an hypercube that might be difficult to illustrate, Fig.1b shows an example of a bigram.

A bigram indicates the number of tuples $[x_1^n(i), x_2^n(i)]$ that lie within ranges of values, called bins b_1, b_2. The frequency of the tuples that falls in each bin, allows the construction of an array with $b_1 \times b_2$ cells. The higher the value in a cell is, the greater the frequency of tuples in it.

So, the complexity of computation of the bigram is equal to the number of tuples to process with the number of bins, such that $O(T \times (B_1 + B_2))$, where B_1 and B_2 are the maximum values to test. For the histogram, we have a complexity of $O(TB)$, where B is the biggest value of the bin.

In our model, we use the bigram tool to reduce the data and extract mutual information inside the MTS. The histogram reduces X^n to a vector of length b, the number of bins, and the bigram reduces it to a matrix of shape $b_1 \times b_2$. Because the Mgram is a statistical model, giving a clue of the density function, it needs a lot of points to do a good assessment of the data. As so, the more we have point per bins, the more accurate it is. So Mgram model might be better for long MTS.

To avoid the lost of order with this representation of time series data, we also include the cumulative sum and the derivative, in addition to the initial data, such that a Mgram is related to the frequencies for initial values, for derivative values, and for the cumulative values.

3 Related work

It exists a plethora of methods about the classification of UTS, among which DTW-1NN [4], [8], SAX [5] and COTE [6] have proven their efficiency.

The *DTW-1NN* is a distance-based classification method. Sakoe and Chiba created the DTW distance to word recognition speeches in 1978. Keogh et al. then adapted it to the UTS classification. It computes the distance between two series taking into account time shift. Until today it's one of the most known and efficient techniques, but may also be time consuming.

SAX is among the transformation-based methods, the one that allows to transform UTS into strings, allowing a reduction by keeping order information and allowing distance measures to be defined on the symbolic approach with lower bounding compared to the original time series distance.

COTE is an ensemble learning method [10] allowing basic and fast classifiers to be combined. So COTE uses a voting scheme on various distance metrics as DTW and UTS transformation as shapelets [18] where a shapelet is an extracted part of a time series. As it is the combination of all the best aspect of UTS classification, it comes out to be more efficient [7].

UTS approaches are not suitable for MTS and [9] reports that their the generalization of UTS methodologies into MTS ones isn't trivial. This is mainly due to the difficulty to take into account the dependence between the MTS dimensions. However, some adaptations exist. We have presented DTW-1NN and SAX because they are part of those methods. For example, *SMTS* is based on SAX [12]. SMTS is a random forest tree algorithm transforming an MTS into a bag of words. WEASEL-MUSE algorithm [13] transforms an MTS into

a histogram of frequency counting of Symbolic Fourier Approximation (SFA) words using multiple sub-windowing series. Those two techniques gives good results, however their complexity aren't suitable to long time series. Indeed, we know that random forests don't scale well with big data as well as with windowing extraction.

Other MTS approaches are part of manifold learning as PCA-Eros [11] or MTSC [15]. They are based on principal component analysis (PCA) [14] which is the most known manifold technique that allows the reduction of the dimensions of an MTS. However one main limitation of manifold learning is its runtime. Another direct method [16] based on a structure of time-based features as the duration and so on. This structure, combined with a new distance, finds the nearest neighbors.

The best methods, based on their accuracy results are WEASEL_MUSE, SMTS and different versions of DTW-1NN called DTW1NND and DTW1NNI presented in [9]. That is why we will compare our results to those methods.

Concerning the Mgram model, it has been used to improve image contrast [19]. This paper describes variations of the used of histograms to determine threshold allowing image contrast enhancement. We also find reference to Mgrams for text contrast in [20]. Finally, since 1998, Michelin used Mgrams to analyze the tire usage [2]. But to our knowledge, this model has never been used for multivariate time series classification. We choose to use the Mgrams model in the context of Multi-view learning [23],[22] as it allows us to extract different views of the data. Multi-view learning applies well to MTS as their dimensions can be seen as different views of the same phenomenon.

4 Ensemble of Mgrams

The base for our work on MTS is the notion of Mgrams. As a simplification of the implementation, explanation, and comprehension, we focus only on bigram and histogram in the application of our model. Algorithm 1 describes it. So the model is an ensemble of bigrams and histograms representing the multiple views of the same data. We extract dimensions of the MTS, created a bigram or histograms and finally we do a voting scheme to do a final prediction.

4.1 Construction of bigrams

As bigram can only deal with two dimensions and histogram with one, we have to pick up dimensions among the MTS. As we cannot choose a priori, which are the dimensions of the MTS containing most of the discriminating information, we choose to do it randomly (Alg 1 row 3.a).

For the bigram construction, as shown in section 2, we have two parameters to tune which are the bins b_1 and b_2 for each dimension. The training complexity is defined by the number of combinations possible of (b_1, b_2) with repetitions because $bigram(b_1 = i, b_2 = j) \neq bigram(b_1 = j, b_2 = i)$. So the complexity of

this training part is $O(B_1 B_2)$ where B_1 and B_2 are the maximum size of bins and $b_1 \in B_1, b_2 \in B_2$.

This time of training can be reduced, even more, be taking $b_1 = b_2$, and so the complexity is $O(B_1)$. In that case with have a square bigram. Concerning the final data storage, the transformation allows to reduce the data to $b_1 \times b_2$ for bigrams and $b_1 + b_2$ for histograms as we compute it also for two dimensions. Those transformations are applied to train and test set and kept in two different lists. Those two lists will be used for the voting scheme classifier, as shown in Fig.2.

4.2 Complement of bigrams

The capacity of bigrams to reduce data, is a huge advantage, but also its main weakness. When we compress data using bigrams, we lost a lot of information as the lengths of events, tendencies, and the orders of events. To get back some of this information, we add two components to our model: the derivative and the cumulative sum of the MTS.

Derivation It‘s easy to find cases where two MTS give the same bigram. If we change the order of the point of an MTS, we get a new MTS that has different tendencies but gives the same bigram. If we consider that those two MTS belong to two different classes, the bigram isn‘t a sufficient tool to classify our data. Therefore, we add to the model the notion of derivative, such that:

$$X‘^n_m = [x^n_m(2) - x^n_m(1), ..., x^n_m(T) - x^n_m(T-1)] \tag{2}$$

When we apply the bigram transformation to the derivatives, we keep information about the tendencies.

Cumulative sum The same reasoning holds for the notion of the order of events by the generalization of blocks of rows. If we change the order of blocks of points, in an MTS, we create a new one where we switch event. But this MTS gives the same bigram. In that case, the derivative isn‘t sufficient. That‘s why we had the cumulative sum to our ensemble, as it is link with integrals and so, point positions. We define it such that:

$$\chi^n_m = [x^n_m(1) + x^n_m(2), ..., \sum_{t=1}^{t=T} x^n_m(t)] \tag{3}$$

When we apply the bigram transformation to the cumulative sums, we keep information about the order of events. Even if, in our case we kept the entire cumulative sum series, the user can only kept few points considering that the information might be drowning in the sum length.

4.3 Final construction

So we have the original MTS, their derivatives, and cumulative sums. As we said earlier, because we implement the bigram version, we extract two dimensions randomly. We re-iterate the process multiple time to get other dimensions (Alg 1 row 3), and we do a voting-scheme classification (Alg 1 row 4). We had the step of correlation dropped to get rid of redundant information (Alg 1 row 2). This allows us to do a fair weight model and faster computation.

Algorithm 1 Ensemble learning with Bigrams and Histograms

Require: : an MTS, bins min max, R number of runs

1. Normalize data (see section 2)
2. Drop correlated dimensions over some threshold
3. k from 1 to R maximum number of runs choose by the user
   (a) Randomly choose 2 dimensions (not already seen) among the M attributes of the MTS
   (b) Randomly choose the transform into bigram or histogram
   (c) Randomly choose to keep, derive, integrate
   (d) Train, transform with the best bin cutting and keep the new data
4. Voting scheme on the set of bigrams and histograms found

The number of re-iteration k is a parameter of the model and defined by the user where the maximum number of run is called R in the model scheme Fig.2. This figure represented the construction of our final model with the concept of multiple views. We see that the bigrams and histograms parameters found on training are applied to the test data. Those bigrams and histograms are then giving their own predictions using learners choose by the users. Finally, we do a voting scheme to get the final prediction.

5 Evaluations

5.1 Experimental Setup

Data The UEA classification website provides 30 datasets about MTS classification [17]. We use those datasets as it is the biggest archive available even if the sizes of the MTS don‘t correspond well to Mgrams computation. As they have also standardized the lengths, we add 20 datasets coming from the Baydogan website [12]. Those datasets have already benchmarked algorithms like WEASEL-MUSE and SMTS. Some datasets are common to the two archives. To conclude, on all the datasets, only *EigenWorms* is in the scope of our work in terms of long lengths. However, it‘s not a variable one. We describe each dataset in a table available on the website extending our work [25].

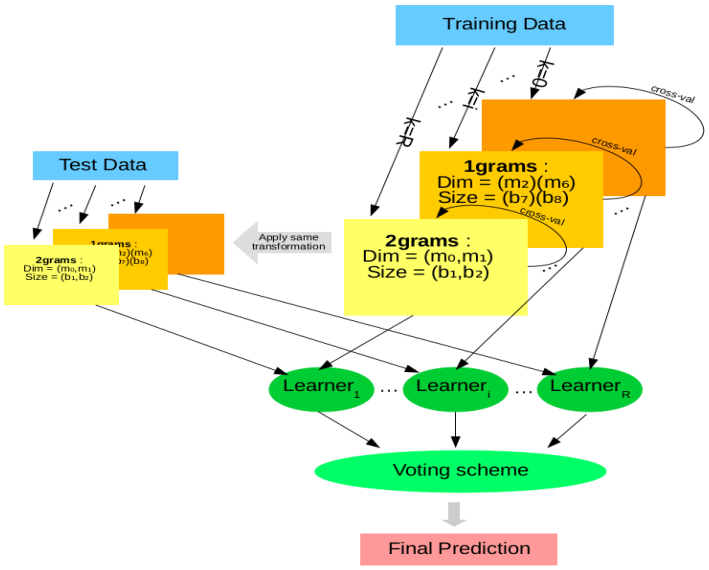


Fig. 2: Scheme of the bigram model

Learners In this publication, the goal isn't to show the ability of the best classifiers but to show what we can accomplish with Mgrams. This is the reason we use only 1-nearest-neighbor classifiers. We use them at each step where we need classification techniques either to find the best bins or the best-weighted voting scheme.

Tuning parameters The parameters to tune are the number of runs k, the size of bins for bigrams b_1, b_2 and histograms b. k is, in our implementation, a percentage of the combinations made by choosing two dimensions among M, transforming or not the MTS (Original, Derivative and cumulative sum), and the tool used (bigram, histogram), giving equation (4).

$$k = percentage \times \binom{2}{M} \times 3 \times 2 \tag{4}$$

We choose two dimensions among the M then we have the choice between three transformations and finally two representations. We try four values for the percentages : $25\%, 50\%, 75\%$ and 95%

For the bins size of bigram, we have $b_1, b_2 \in [2, 11]$ and $[b \in [2, 50]$ for histogram. The bounds chosen are based on the fact that allowing more bins, only add zeros in the matrices of the bigrams and histograms compared to the size of the MTS available.

Also, in our case, we get rid of the range parameters, presented in Section 2, using the min-max normalization. Doing so, we have the data in range $[0, 1]$.

Otherwise, the user can choose the normalization methods of its choice, but got another parameter to tune.

5.2 Main properties

First, we will expose the main properties of the Mgrams tools, show the influence of parameters on the models, as its robustness or its capacity to deal with variable lengths of MTS. Also we will describe special applications of our model with its square version and its complete Mgram one.

Robustness As we explain in section 5.1, we have a parameter k defined by a percentage of combinations. We tried the percentage $P \in [25\%, 50\%, 75\%, 95\%]$. We show here, that even if we execute a small amount of the combination, *i.e.* 25%, we have robust results.

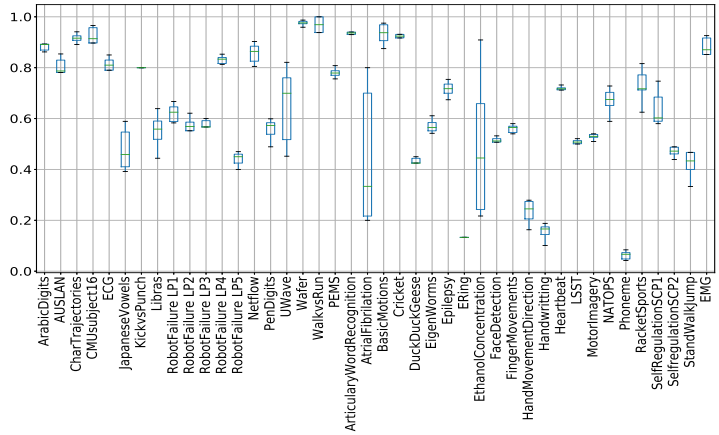


Fig. 3: Box plot of standard deviation and mean of accuracies for 25% of the combination

We see in Fig.3 that for only a few datasets, the standard deviation lift. Otherwise, we have small variations. So we can conclude that most of the dimension of a MTS contain enough information to classify. Our model with 25% of the total combinations is robust enough for most of the datasets. The other dataset are special cases where some dimensions didn't contain the useful classification information. So when we extract their bigrams, they aren't sufficient to classify the data.

We also show in Fig.4 that we didn't have compute all the bigrams, histograms to perform well. The mean and the standard deviation of the box plot are smaller for 25% than for 95%. We can conclude that adding too much information is equal to adding noise into the process. So it's better, for computation

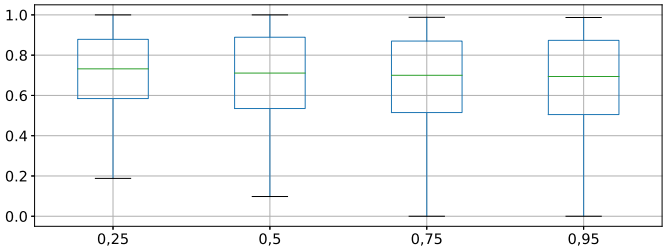


Fig. 4: Box plot of standard deviation and mean of accuracies among all the dataset for 25%, 50%, 75%, 95%

time consuming and accuracy, to take randomly a small part of dimensions instead of all.

Dealing with length In Section 2, we explained that the performances of our model are linked to the length of the time series. As bigram is a statistic tool, the number of points available impacts its effectiveness.

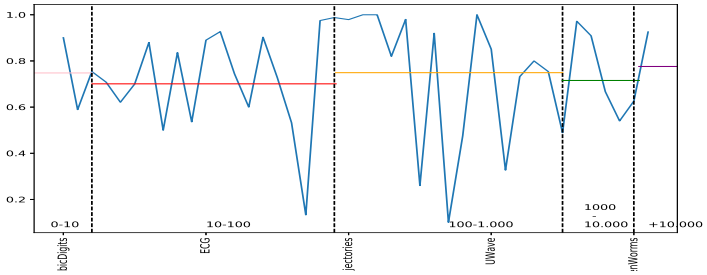


Fig. 5: Accuracies by size, with mean of accuracies by size-interval

We can see in Fig.5 that there is a correlation between the size of the time series and the performances of our model, as the mean of the accuracies grows with the size of the dataset.

Activities Also, we want to see if there might exist domains of activities where our algorithm performs better than others. We regroup activities as in [13] categories: Handwriting, Motions, Sensors and Speech recognition.

We can't see any tendency in Fig. 6 to make a clear conclusion about activities where our algorithm performs better. So, we have shown that the bigram can be a useful tool to extract, reduce and classify data and that there is no impact concerning the activities type of dataset.

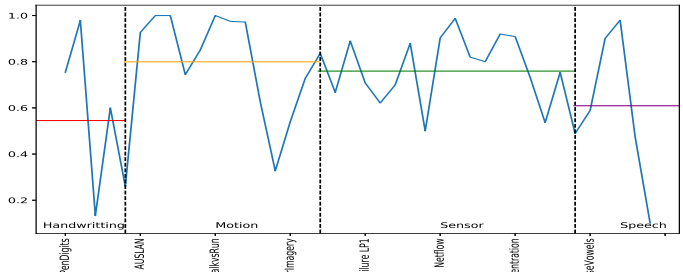


Fig. 6: Accuracies by domain of dataset

5.3 Mgrams

We present here one of the special application of our model. An M-gram means we combine all the M dimensions of the MTS together. We create an M-array compiling the frequency of appearance per interval of each tuple $[x_1^n(i), x_M^n(i)]$. We are using them with the same model established for bigram. We replace bigrams by M-grams in Algorithm 1. First, this model doesn't scale well to high-dimensional data in terms of space needed and time of execution. Then, objects with over three dimensions are hard to handle and even more to represent. We only test datasets where $M \leq 8$ which is already a large object. For example if take bins of size 2, we have a array of 256 cells. The size of a M-grams is define by $\prod_{i=1}^{M} b_i$. Also the complexity is $O(T(\sum_{i=1}^{M} b_i))$. The results presented in Table 1 load to two conclusions. The first is that studying all the dimensions of an MTS is unnecessary as the M-grams model performs less than a simpler bigrams model as shown in Fig.7. And as it takes more time and space, it's not a direction to study in our future work. The second is that some dimensions are noisy as all combined dimension doesn't imply better results that combinations of some. This result confirm the robustness of our model present before.

5.4 Square bigrams

During the choice of the bins combination, one way to reduce the number of computations is to take only equal bins for the two dimensions. Meaning, we have $b_1 = b_2$, so instead of trying $B_1 \times B_2$ combinations, we try only B_1. It allows a huge reduction in the training times. This might be a compromise for large datasets, even if we lost accuracy in prediction. We see Fig.7 that this model is still preferable to the Mgrams model.

5.5 SAX and bigrams

It seems to be important to test if combining bigrams with other time series classification techniques may give interesting results. Therefore, we try this combination with the SAX method. We keep our model as in Fig.2 and replace histogram

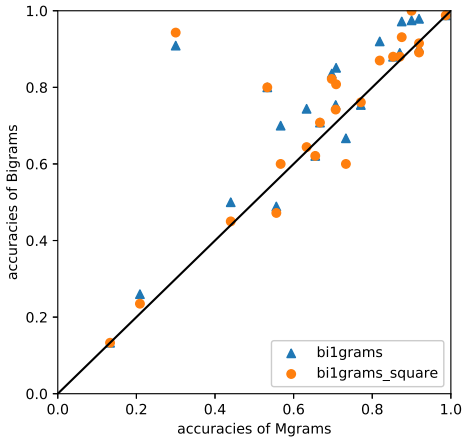


Fig. 7: The accuracy-projection of the bigrams models and the square one depending of the Mgrams model. Points above the $x = y$ line mean bigrams and square bigrams models are better than the Mgrams one.

by SAX transformation. The goal here is to show that bigram is a way to apply UTS methods to MTS datasets.

SAX The SAX algorithm was created in 2007 by *Lin et al.* [5]. During the experimentations, we used to tune the number of segments and kept an alphabet of 10 digits from 0 to 9. It allows us to replace easily histograms vectors by SAX ones. Otherwise, the model scheme is the same as described in Section 4. We didn't apply the distance presented in [5] to allow an easier integration to our model. This application is possible because SAX is still in the euclidean space.

Combination As seen in the critical diagram Fig.8, the Bigram-SAX model performs better than the SAX one and there is no critical difference between our previous model and this new one.

We can conclude that the combination of SAX with Bigrams improves the performs of the algorithm lonely. Therefore, Bigrams can be a complement of UTS classification algorithms to an MTS application.

5.6 Accuracy on benchmark data

Finally, we do a global comparison of our methods with other MTS methods. We applied our models, called $B1gr1NN$ and $BSAX1NN$ in table 1, and compared it to $SMTS$ [12], $WEASEL\_MUSE$ [13], $DTW1NN_I$, $DTW1NN_D$ [17] as described in Section 3. We also have $ED1NN$ as a benchmark algorithm also

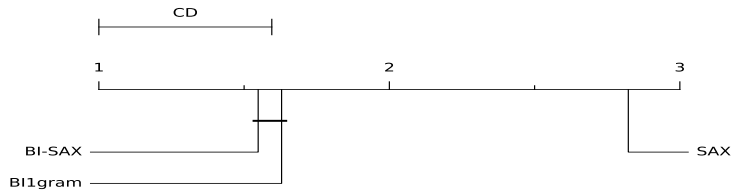


Fig. 8: Critical difference diagram on average ranks for the bigrams models and the bigrams-SAX one.

present in [17], which the simplest nearest neighbors algorithm based on the Euclidean distance. For all the competing algorithms, we get their results from their original publications, as we didn't have the best tuning parameters for each methods to compute them ourself. The results presented for our models are the best of all tuning, get through multiple cross-validation on train data. The models $SAX1NN$ and $BSAX1NN$ correspond to the application of SAX and SAX with our model. The last models $SquarB1gr1NN$ is the application of a square bigram and $M1gr1NN$ to the Mgrams one.

We can already see that our model can perfectly compete with $DTW1NN_I$, $DTW1NN_D$ in Fig.9a. We do two critical diagrams because we didn't get the accuracy results on the same datasets for all the models. However, in Fig.9b we see the two benchmarks models perform better than us but their is no critical difference. Indeed, excepted Japanese Vowels dataset, where we perform bad, our other results are closed to $SMTS$ and $WEASEL\_MUSE$ ones. Also as explained in Section 4., the complexity of our model is better than random forest and windowing extraction techniques ones, making our model faster. For the *JapaneseVowels* we perform bad because this is a short length dataset with many dimensions. However as it's supposed to, we perform better with long time series. This is precisely why we use Mgram. The more it reduces the size of the MTS, the more effective it is.

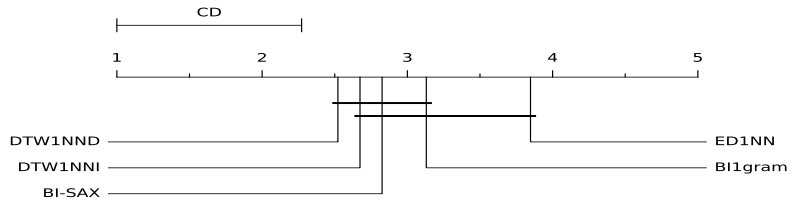
6 Conclusion

The fast exploitation of large MTS, as the compact representation of variable long time series, are challenges we face here. The Mgrams model allows to reduce original data to a more compact representation of data. We show that its bigrams version gives good results even if we are not exhaustive in terms of combinations of dimensions and transformations. It's also a visual object allowing a faster comprehension for humans. Finally, we demonstrate that our model is better with long MTS and can handle time series with variable lengths between dimensions. The bigram models can be combined with other UTS classification methods.

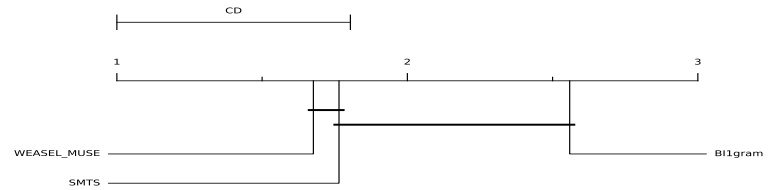
Our approach is based on a random choice of bigrams for multi-view learning, and this is a point of improvement of our method.

	$ED1NN$	$DTW1NN_I$	$DTW1NN_D$	$SMTS$	$WEASEL\_MUSE$	$B1gr1NN$	$SquarB1gr1NN$	$M1gr1NN$	$SAX1NN$	$BSAX1NN$
AUSLAN				0.947	**0.97**	0.927	0.902			
CharTrajectories	0.964	0.969	0.989	**0.992**	0.973	0.979	0.915	0.919		
CMUsubject16				**1.0**	**1.0**	**1.0**	**1.0**			
ECG				0.818	0.88	**0.89**	0.88	0.87		
JapaneseVowels	0.924	0.959	0.949	0.969	**0.976**	0.589	0.535			
KickvsPunch				0.82	**1.0**	**1.0**	0.8			
Libras	0.833	0.894	0.87	**0.909**	0.894	0.744	0.644	0.633	0.656	0.794
RobotFailureLP1				0.856	**0.94**	0.708	0.708	0.667	0.50	0.688
RobotFailureLP2				**0.76**	0.733	0.621	0.621	0.655	0.517	0.69
RobotFailureLP3				0.76	**0.9**	0.7	0.60	0.567	0.5	0.6
RobotFailureLP4				0.895	**0.96**	0.88	0.88	0.853	0.453	0.8
RobotFailureLP5				0.65	**0.69**	0.5	0.45	0.44	0.44	0.52
NetFlow				**0.977**	0.961	0.903	0.891	0.919		
PenDigits	0.973	0.939	**0.977**	0.917	0.912	0.754	0.742	0.707	0.785	0.818
UWave	0.881	0.868	0.903	**0.941**	0.916	0.851	0.808	0.708	0.791	0.878
Wafer				0.965	**0.997**	0.988	0.988	0.987		
WalkvsRun				**1.0**	**1.0**	**1.0**	**1.0**			
ArabicDigits	0.967	0.959	0.963	0.964	**0.992**	0.901	0.90			
ArticularyWordRecognition	0.97	0.98	**0.987**			0.98	0.96		0.677	0.98
AtrialFibrilation	0.267	0.267	0.22			**0.8**	**0.8**	0.533	0.467	**0.8**
BasicMotions	0.676	**1.0**	0.975			**1.0**	**1.0**	0.9	0.425	0.95
Cricket	0.944	0.986	**1.0**			0.972	0.931	0.875	0.472	**1.0**
DuckDuckGeese	0.275	0.55	**0.6**			0.475	**0.45**		0.2	0.2
EigenWorms	0.549	-	0.618			0.626	0	**0.718**	0.291	0.58
Epilepsy	0.666	**0.978**	0.964			0.92	0.87	0.819	0.623	0.90
EthanolConcentration	0.293	0.304	0.323			**0.943**	**0.943**	0.304	0.285	0.654
Ering	0.133	0.133	0.133			0.133	0.133	0.133	**0.3**	0.133
FaceDetection	0.519	-	0.529			**0.532**	0.511		0.503	0.517
FingerMovements	0.55	0.52	0.53			**0.6**	0.59		0.49	**0.6**
HandMovementDirection	0.278	0.306	0.231			0.33	0.33		0.365	**0.381**
HandWriting	0.2	**0.316**	0.286			0.26	0.235	0.209	0.126	0.228
Heartbeat	0.619	0.658	0.717			0.732	0.727		0.722	0.727
LSST	0.456	0.575	0.551			0.536	0.535		0.441	0.539
MotorImagery	0.51		0.5			0.56	0.56	0.440	0.5	**0.56**
NATOPS	0.85	0.85	0.883			0.728	0.733		0.594	0.767
PEMS	0.705	0.734	0.711			0.82	0.792		0.792	**0.827**
Phoneme	0.104	0.151	0.151			0.101	0.96		0.094	0.094
RacketSports	**0.868**	0.842	0.803			0.836	0.822	0.697	0.487	0.783
SelfRegulationSCP1	0.771	0.765	0.775			0.761	0.761	0.771	0.744	0.812
SelfRegulationSCP2	0.483	0.533	0.5339			0.489	0.472	0.556	544	**0.583**
StandWalkJump	0.2	0.333	0.2			**0.667**	0.6	0.733	0.467	**0.667**

Table 1: Accuracies on benchmark dataset.
The *light grey* corresponds to the results we don't have because they haven't been published yet. Whereas, the *dark gray* corresponds to the results we didn't have because we can't compute them.For the Bi-Sax method and the SAX, it is because they are incompatible with variable length MTS (see Section 6.3). For Mgrams, it is because of the number of dimensions incompatible with an exhaustive combination of them (see Section 6.6).



(a) Critical difference diagram on MTS with fixed lengths to compare our models with ED1NN, $DTW1NN_D$ and $DTW1NN_I$



(b) Critical difference diagram on MTS with variable lengths to compare our models with $SMTS$ and $WEASEL\_MUSE$

Fig. 9: Critical difference diagram of our models compared to benchmarks.

References

1. Ordez, P., DesJardins, M., Feltes, C., Lehmann, C.U., Fackler, J.: Visualizing multivariate time series data to detect specific medical conditions. AMIA Annu Symp Proc, pp. 530–534 (2008)
2. Le Maître, O., Sssner, M., Zarak, C.: Evaluation of Tire Wear Performance. SAE Technical Paper, SAE International, https://doi.org/10.4271/980256 (1998)
3. Zbigniew, J., Holger, Z.: The DEBS 2014 Grand Challenge. Proceedings of the 2014 ACM International Conference on Distributed Event-based Systems, pp. 266–269 (2014)
4. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing **26**(1), pp. 43–49 (1978)
5. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Mining and Knowledge Discovery **15**(2), pp. 107–144 (2007)
6. Bagnall, A., Lines, J., Hills, J., Bostrom, A.: Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. IEEE Transactions on Knowledge and Data Engineering **27**(9), pp. 2522–2535 (2015)
7. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **31**(3), pp. 606–660 (2017)
8. Keogh, E., Pazzani, M.: Derivative Dynamic Time Warping. Proceedings of the 2001 SIAM International Conference on Data Mining, pp. 1–11 (2001)

9. Shokoohi-Yekta, M., Wang, J., Keogh, E.: On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case. Proceedings of the 2015 SIAM International Conference on Data Mining, pp. 289–297 (2015)
10. Breiman, L.: Bagging Predictors. Machine Learning **24**(2), pp. 123–140(1996)
11. Yang, K., Shahabi, C.: A PCA-based Similarity Measure for Multivariate Time Series. Proceedings of the 2Nd ACM International Workshop on Multimedia Databases, pp. 65–74 (2004)
12. Baydogan, M., Runger, G.: Learning a symbolic representation for multivariate time series classification. Data Mining and Knowledge Discovery **29**(2), pp. 400–422 (2015)
13. Schäfer, P., Leser, U.: Multivariate Time Series Classification with WEASEL+MUSE. CoRR, https://arxiv.org/abs/1711.11343 (2017)
14. Jolliffe, I., Cadima, J.: Principal component analysis: A review and recent developments. Philosophical transactions Series A Mathematical physical and engineering sciences **374**(2065) pp. 20150202 (2016)
15. Zhou, PY., Chan, K.: A Feature Extraction Method for Multivariate Time Series Classification Using Temporal Patterns. Advances in Knowledge Discovery and Data Mining, pp. 409–421 (2015)
16. Kadous, M., Sammut, C.: Classification of Multivariate Time Series and Structured Data Using Constructive Induction. Machine Learning **58**(2), pp. 179–216 (2005)
17. Bagnall, A., Dau, H., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Eamonn K.: The UEA multivariate time series classification archive. CoRR, https://arxiv.org/abs/1811.00075 (2018)
18. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 947–956 (2009)
19. Shah, G., Khan, A., Shah, A., Raza, M., Sharif, M.: A review on image contrast enhancement techniques using histogram equalization. Science International, **29**, pp. 1297–1302 (2015)
20. Tan, C., Wang, Y., Lee, C: The use of bigrams to enhance text categorization. Information Processing and Management, **38**(4), pp. 529–546 (2002)
21. Yi, B.K., Faloutsos, C.: Fast time sequence indexing for arbitrary lp norms. Proceedings of the 26th international conference on very large databases **10**(14), pp. 385–394 (2000)
22. Yin, Q., Wu, S., Wang, L.: Unified subspace learning for incomplete and unlabeled multi-view data. Pattern Recognition, **67**, pp. 313 – 327 (2017)
23. Zhao, Q., Xie, X., Xu, X., Sun, S.: Multi-view learning overview: Recent progress and new challenges. Information Fusion, **38**, pp. 43 – 54 (2017)
24. Cerqueira V., Torgo L., Pinto F., Soares C.: Arbitrated Ensemble for Time Series Forecasting. In: Ceci M., Hollmn J., Todorovski L., Vens C., Deroski S. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2017. Lecture Notes in Computer Science, vol 10535. Springer, Cham (2017)
25. Homepage publication, https://sites.google.com/view/mts-bihistograms/accueil (March 2019).