SAXO : An Optimized Data-driven Symbolic Representation of Time Series

A. Bondu, M. Boullé, B. Grossin

***Abstract*— In France, the currently emerging *"smart grid"* and more particularly the 35 millions of *"smart meters"* will produce a large amount of daily updated metering data. The main french provider of electricity *(EDF)* is interested by compact and generic representations of time series which allow to accelerate the processing of data. This article proposes a new data-driven symbolic representation of time series named SAXO, where each symbol represents a typical distribution of data points.Furthermore, the time dimension is optimally discretized into intervals by using a parameter free Bayesian coclustering approach *(MODL)*. SAXO is favorably compared with the SAX representation by evaluating a classifier trained from recoded datasets. Our experiments highlight a significant gap in performance between both approaches.**

I. Introduction

The french electrical grid is on the point of being modernized by exploiting information and communications technologies. The emerging *"smart grid"* has multiple objectives : i) the control of the grid and the quality of the electricity supply have to be optimized, despite the fact that power stations are highly distributed; ii) the production of electricity has to be scheduled by taking into account the uncertainty related to renewable energy *(ex: wind, sun exposure)*; iii) the electrical demand need to be coordinated to flatten the consumption peaks and to limit their impact on the environment.

The *"smart meters"* constitute the bottom of the smart grid hierarchy. These new digital meters will be set up in all french households, within a few years. Smart meters are able to record the individual power consumptions in real time, and to send this information to a data center through a communication network. Currently, all technical choices are not yet finalized, but we can reasonably assume that the recorded time series will be sampled every 30 minutes for the 35 millions of smart meters.

The needs of EDF[1] in supervised and unsupervised analysis are increasingly important. For instance, a clustering algorithm may reveal unexpected behaviors from the consumptions of our customers [1, 2]. Supervised approaches *(i.e. classification, regression and forecasting)* can be exploited in many application areas. The most obvious example is the forecasting of the electricity demand which allows EDF to schedule the production of the power stations for the next day [3].

In practice, this kind of data is difficult to analyze efficiently because of : i) the large size of datasets; ii) the inherently high dimensionality of time series; iii) the volatility of individual electrical consumptions. Therefore, the question of the representation of time series becomes crucial.

Our objective is to find a **generic**, **compact**, **informative** and **adaptive** representation of time series. In the ideal case, a generic representation should be exploited by a large range of Data Mining and Machine Learning algorithms. A compact and informative representation should reduce the dimensionality of time series, while limiting the loss of information. More precisely, the processing of the recoded datasets could be drastically accelerated and the quality of our analysis could be improved at the same time. Furthermore, the storage of this kind of data could be facilitated by compressing the initial datasets. An adaptive representation should be driven by the data in order to maximize the preserved information, it should be theoretically grounded and parameter-free.

This article proposes a new data driven symbolic representation of time series. Section II deals with related work. More particularly, the SAX approach [4] which is a commonly used symbolic representation is presented in details. Section III presents the new SAXO approach which is based on a Bayesian co-clustering method. The SAXO representation is first learned from a sampled dataset, and then deployed on the entire dataset. These two steps allow to process massive datasets by using algorithms with appropriated time complexity. In Section IV, our approach is favorably compared with the SAX approach. These comparative experiments are carried out on public supervised datasets, by applying a classifier on several versions of the recoded time series. At last, perspective and future works are discussed in Section V

II. Related work

In comparison with the other types of data, time series are particularly difficult to treat due to their inherently high dimensionality [5, 6]. The curse of dimensionality [7] states that, for a given statistical analysis and for a fixed accuracy, the number of training examples must increase exponentially with the dimension of the data. Virtually all data mining algorithms scale poorly with the dimensionality. During the two last decades, numerous high level representations of time series have been proposed to overcome this issue. The most commonly used approaches are : the Discrete Fourier Transform [8], the Discrete Wavelet Transform [9, 10], the

Alexis Bondu is with EDF R&D, 1 avenue du Général de Gaulle 92140 Clamart France. http://alexisbondu.free.fr

Marc Boullé is with Orange Labs, 2 avenue Pierre Marzin, 22300 Lannion France. http://perso.rd.francetelecom.fr/boulle/

Benoit Grossin is with EDF R&D, 1 avenue du Général de Gaulle 92140 Clamart France.

[1]EDF *(Electricité de France)* is the main french provider of electricity.

Discrete Cosine Transform [11], the Piecewise Aggregate Approximation [12].

Such approaches can be exploited, as a preprocessing, to recode a collection of time series. Then, a data mining algorithm can be applied on the recoded dataset. The main challenge is to improve the scalability of the data mining algorithms, while maximizing the quality of the performed analysis. A good trade-off must be reached between the compression of the input dataset and the preserved information. A data driven representation of time series could be helpful to reach this objective.

Notations for time series :
In this article, the input dataset $\mathcal{D}$ is considered to be a collection of N time series S_i (with $i \in [1, N]$). Each time series consists of m_i data points, which are couples of values X and timestamps T. The total number of data points is denoted by $m = \sum_{i=1}^{N} m_i$. As presented below, a symbolic representation discretizes the series into w time intervals and symbolizes it by using an alphabet of α symbols.

A. Overview of SAX

The symbolic representations consist in discretizing the original time series into symbolic strings [4]. A large amount of algorithms and data structures coming from text mining can be applied on such recoded data [13, 14]. Numerous papers deal with the symbolic representation of times series [15], the SAX *(Symbolic Aggregate approXimation)* approach has the particularity of providing a distance measure that lower bounds the euclidian distance defined in the original space of the time series.

The SAX representation aggregates values within time intervals, which reduces the dimension of the recoded dataset. SAX also provides a *"numerosity reduction"* by discretizing the average values of each time interval by symbols. The number of distinct forms of symbolic time series is drastically reduced. Actually, the SAX representation allows to highly compress the time series and drastically accelerates the applied data mining algorithms.

The SAX representation has become an essential tool for time series data mining. This approach has been exploited to implement various tasks on large time series datasets, including similarity clustering [16], anomaly detection [17, 18], discovery of motifs [19], visualization [20], stream processing [4]. Originally, the SAX approach was designed for indexing very large sets of signals [21], and is still a reference in this field.

B. General principle of SAX

The SAX representation has been introduced by J. Lin and al [4]. This approach consists in two successive steps : i) transform the original time series into Piecewise Aggregate Approximation (PAA) [22, 23]; ii) symbolize the PAA representation into a discrete string. In the particular case of the SAX representation, we consider that all time series have the same length, denoted by m_*.

PAA transform :
An original time series $S = \{x_1, ..., x_{m_*}\}$ is represented by a vector $\bar{S} = \{\bar{x_1}, ..., \bar{x_w}\}$ of w mean values calculated on regular time intervals. More precisely, the *ith* element of $\bar{s}$ is defined as follows :

$$\bar{x_i} = \frac{w}{m_*} \sum_{j=\frac{m_*}{w}(i-1)+1}^{\frac{m_*}{w}i} x_j$$

The time series is divided into w equal size time intervals, that allows to represent a time series S in a w-dimensional space. The parameter w is arbitrarily fixed. Figure 1 is extracted from [4] and plots an example of time series associated with its PAA representation.

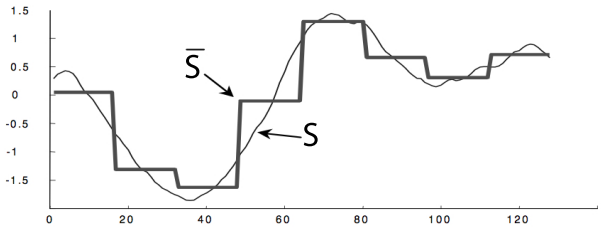

Fig. 1. Example of PAA representation.

Before being converted into the PAA representation, each time series is independently normalized *(the mean equals to zero, the standard deviation equals to one)*. This preprocessing aims at encoding the "shape" of times series, and then compare time series with different offsets and magnitudes [24].

From PAA to symbolic representation :
The SAX transform aims at discretizing the mean values $\{\bar{x_i}\}$ obtained from the PAA transform, into a set of α equiprobable symbols. The interval of values corresponding to each symbol can be analytically calculated under the assumption that the time series have a Gaussian distribution [4]. An alternative method consists in empirically calculating the quantiles of values in the dataset.

Figure 2 plots an example of SAX transform based on a set of three symbols *(i.e. $\{a, b, c\}$)*. The left part of this figure illustrates that the distribution of values is supposed to be known, and is divided into equiprobable intervals corresponding to each symbol. Then, these intervals are exploited to discretize the mean values into symbols within each time interval. The concatenation of symbols ***baabccbc*** constitutes the SAX representation of the original time series, also called a SAX word.

C. Limitations of SAX

The SAX representation appears to be really helpful for processing large datasets of time series, owing to dimensionality and numerosity reduction. However, this approach suffers from several significant limitations.

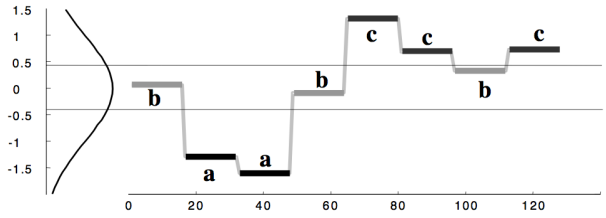

Fig. 2. Example of SAX representation.

A lossy compression approach :
SAX is a surjective transform : there is an infinity of time series matching with a fixed SAX word. The set of time series compatible with an arbitrary chosen SAX word gathers a large variety of time series, including extremely noisy and smooth time series. Figure 3 illustrates this point by plotting two time series represented by the same SAX word.

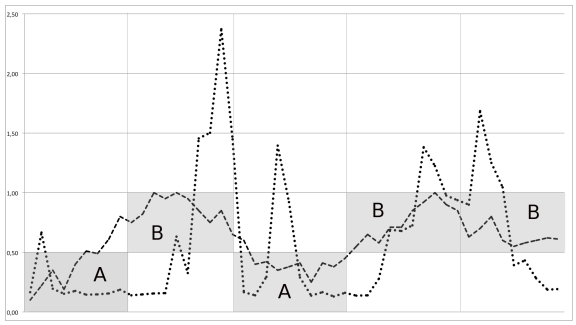

Fig. 3. Example of two time series represented by the same SAX word : one of these is smooth, and the other one is very noisy.

In practice this phenomenon can be problematic, since most of the information carried by the dataset may be lost. For instance, a clustering algorithm applied on a SAX re-encoded dataset may not find interesting groups due to this loss of information. The SAX representation can be viewed as a lossy compression approach. The lost information cannot be quantified in the general case *(i.e. without prior hypothesis on data)*. The fact that each symbol represents an interval of average values calculated within a time interval appears to be low informative. We claim that the loss of information could be reduced and regularized by the dataset. In this paper, the proposed approach associates a typical distribution of data points to each symbol.

The discretization is not driven by the data :
The SAX transform involves : i) a discretization of "time" into equal size intervals; ii) a discretization of "values" into equiprobable symbols. As shown on Figure 3, these two partitions constitute a two-dimensional grid that is exploited to re-encode the time series. The SAX approach carries out this discretization independently of the dataset, and both dimensions *("time" and "values")* are considered to be independent. We claim that an efficient discretization approach may improve the SAX representation, especially by catching the joint density of "time" and "values". In that way, we aim at reducing the lost information due to the compression of data.

The alphabet has the same meaning over time :
In the SAX representation, each symbol has the same meaning over time whatever its rank in the SAX words. This lack is common to all existing symbolic representations, which assume that the distribution of data points is constant over time. For instance, the input dataset may represent the daily electricity consumptions of households. The variety of the consumer's behavior is not the same during the night than during daytime, and cannot be optimally encoded by the same alphabet. We claim that the dependency between data points and time may be caught by providing a different meaning to the symbols over time.

Two parameters need to be adjusted :
The SAX transform involves two user parameters *(i.e. the number of time intervals w and size of the alphabet α)*. The impact of these parameters on the quality of subsequently analysis is difficult to evaluate in practice.

III. SAXO : A NEW DATA DRIVEN APPROACH

This paper proposes a free parameter approach, which optimizes the following modeling choices in a fully data driven way : i) the number of time intervals; ii) the bounds of time intervals; iii) the typical distributions of data points associated with each symbol, into each time interval.

A. Overview

The SAXO approach *(Symbolic Aggregate approXimation by data-driven Optimization)* is designed to overcome the limitations of existing symbolic representations *(see Section II-C)*. This approach optimizes the discretization of *time* and *values*. Each symbol matches with a *typical distribution* of data points, which depends on the rank of the symbol.

The SAXO approach widely exploits the unsupervised discretization method MODL [25], our choice is motivated by the good statistical properties of this approach :

- MODL is **theoretically grounded** and exploits an objective Bayesian approach [26] which turns the discretization problem into a task of model selection. The Bayes formula is applied by using a low-informative prior distribution and leads to an analytical criterion which represents the probability of a model given the data. Then, this criterion is optimized in order to find the most probable model given the data. The number of intervals and their bounds are automatically chosen.
- The best discretization model **estimates the joint density** of explicative variables by a *multidimensional data grid*.
- MODL is a **nonparametric approach** in the meaning of C. Robert [26] : the number of modeling parameters increases continuously with the number of training

examples. Any joint distribution can be estimated, provided that enough examples are available.

The SAXO approach aims at processing very large datasets of time series, and exploits two successive algorithms. First, the learning phase consists in defining the representation from a sampled dataset with a $\mathcal{O}(m\sqrt{m}\log m)$ time complexity *(Algorithm 1, Section III-C)* . Then, the SAXO representation is efficiently deployed on the entire dataset with a $\mathcal{O}(m)$ time complexity *(Algorithm 2, Section III-C)*.

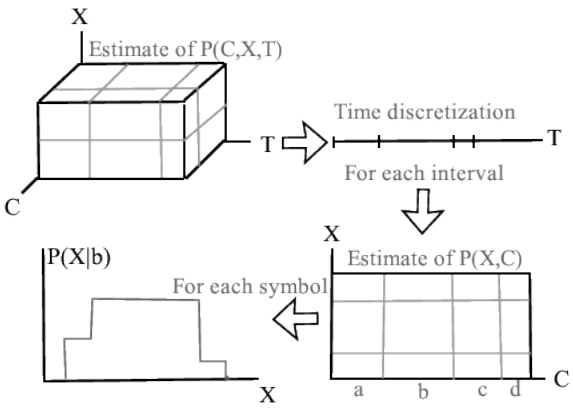

Fig. 4. Main steps of the learning step

Figure 4 gives the intuition of the SAXO approach by illustrating the main steps of the learning algorithm. The joint density $P(C, T, X)$ of the IDs of time series C, the values X, and the time T is estimated by a trivariate coclustering model. The outcome *time* discretization is retained, and the joint density $P(C, X)$ is estimated within each time interval by using a bivariate coclustering model. The resulting clusters of time series are characterized by piecewise constant distributions of values, and corresponds to the symbols.

Section III-B presents in details the MODL coclustering approach and explains how this approach has been adapted to the particular case of time series. Section III-C is dedicated to the SAXO algorithms which learn the representation from data and deploy it on the entire dataset.

B. MODL : Clustering approach for times series

This section summarizes the MODL *(Minimum Optimized Description Length)* coclustering approach and its application to the clustering of time series detailed in [27].

Data Grid Models :

The MODL coclustering approach allows to automatically estimate the joint density of several *(numerical or categorial)* variables, by using a data grid model [25]. A data grid model consists in partitioning each numerical variable into intervals, and each categorical variable into groups. The cross-product of the univariate partitions constitutes a data grid model, which can be interpreted as a nonparametric piecewise constant estimator of the joint density. A Bayesian approach selects the most probable model given the dataset, within a family of data grid models.

Application to time series :

Each time series consists of m_i data points, which are characterized by two variables : T represents the timestamps and X is the values of the data points. The *ith* time series is denoted by $S_i = (t_{ij}, x_{ij})_{j=1}^{m_i}$. The MODL approach handles a re-encoded dataset $\mathcal{P}$ which contains the m data points of $\mathcal{D}$ in a tabular format. Each data point is characterized by three variables : C represents the "id" of the original time series, T and X.

Then, the coclustering approach is applied to estimate the joint density $P(C, T, X)$ by a trivariate coclustering model. As $P(T, X|C) = \frac{P(C,T,X)}{P(C)}$, this model can also be interpreted as an estimator of the joint density between T and X, which is constant within each cluster of time series. In others words, this clustering approach gathers time series with respect to the joint density of their data points. This property is particularly interesting when the handled time series are noisy.

A clustering model M is defined by :

- a number of clusters of time series;
- a number of intervals for each data point dimension *(time and values)*;
- the partition of the time series into the clusters;
- the distribution of the data points on the cells of the data grid;
- for each cluster, the distribution of the data points on the time series belonging to the same cluster.

Notations for trivariate coclustering :

- k_C *: number of clusters of time series;*
- k_T, k_X *: number of intervals of time and values;*
- $k = k_C k_T k_X$ *: number of cells of the data grid;*
- $k_C(i)$ *: index of the cluster that contains the series* S_i*;*
- $\{n_{i_C}\}$ *: number of time series within each cluster* i_C*;*
- $\{m_i\}$ *: number of data points of each time series* S_i*;*
- $\{m_{i_C}\}$ *: number of data points within each cluster* i_C*;*
- $\{m_{j_T}\}$ *: number of data points in the time intervals* j_T*;*
- $\{m_{j_X}\}$ *: number of data points in the intervals* j_X*;*
- $\{m_{i_C j_T j_X}\}$ *: number of data points belonging to each cell* (i_C, j_T, j_X).

The MAP model *(maximum a posteriori)* is selected by a Bayesian approach within $\mathbb{M}$, the set of all possible coclustering models. The MAP maximizes the product of the prior distribution $P(M)$ and the likelihood of data given the model $P(D|M)$. The exploited prior distribution $P(M)$ is derived from the minimum length description principle. The prior for the parameters of clustering model are chosen hierarchically and uniformly at each level :

- the numbers of clusters k_C and of intervals k_T, k_X are independent from each other, and uniformly distributed between 1 and N for time series, and between 1 and m for the point dimensions;

- for a given number k_C of clusters, every partition of the N time series into k_C clusters are equiprobable;
- for a given model size (k_C, k_T, k_X), every distribution of the m data points on the k cells are equiprobable;
- within a given cluster of time series, every distribution of the points on the time series belonging to the same cluster are equiprobable;
- for a given interval of T *[resp. X]*, every distribution of the ranks of the T *[resp. X]* values of points are equiprobable.

Taking the negative logarithm of $P(M)P(D|M)$, a clustering model M is Bayes optimal if the value of the following criteria is minimal :

$$c(M) = \log N + 2\log m + \log B(N, k_C) \tag{1}$$

$$+\log\binom{m+k-1}{k-1} + \sum_{i_C=1}^{k_C} \log\binom{m_{i_C}+n_{i_C}-1}{n_{i_C}-1}$$

$$+\log m! - \sum_{i_C=1}^{k_C}\sum_{j_T=1}^{k_T}\sum_{j_X=1}^{k_X} \log m_{i_C j_T j_X}!$$

$$+\sum_{i_C=1}^{k_C}\log m_{i_C}! - \sum_{i=1}^{N}\log m_i! + \sum_{j_T=1}^{k_T}\log m_{j_T}! + \sum_{j_X=1}^{k_X}\log m_{j_X}!$$

The two first lines of Equation 1 correspond to the prior distribution $P(M)$. The terms "$\log N$" and "$2\log m$" relate to the prior distribution of the numbers of clusters and intervals. This means that the probability of observing a particular value of k_C *[resp. k_T and k_X]* is supposed to be $1/N$ *[resp. $1/m$]*. The probability of a given partition of the time series into clusters is given by $1/B(N,k)$, where $B(N,k)$ is the number of possible ways of partitioning a set of N elements into k subsets, eventually empty [2]. The prior probability of observing a given multinomial distributions of the m data points on the k cells of the data grid is given by $1/\binom{m+k-1}{k-1}$. At last, the prior term $1/\binom{m_{i_C}+n_{i_C}-1}{n_{i_C}-1}$ represents the probability of observing a particular multinomial distribution of the data points of the cluster i_C, on the time series belonging to the same cluster. The two last lines of Equation 1 correspond to the likelihood of data $P(D|M)$. The third line stands for the likelihood of the distribution of the data points on the cells. The last line corresponds to the likelihood of the distribution of the points of each cluster on the time series of the cluster, followed by the likelihood of the distribution of the ranks of the T values *[resp. X values]* in each interval.

An efficient algorithm based on a greedy heuristic and a neighborhood exploration has been implemented to optimize the evaluation criterion [25]. This algorithm is super linear and finds a good discretization model within a $\mathcal{O}(m\sqrt{m}\log m)$ time complexity. This algorithm can be constrained by specifying a maximum number of clusters of time series, or a maximum number of intervals. This feature is exploited in Section IV for a better interpretability of extracted patterns.

[2]More precisely, $B(N,k) = \sum_{i=1}^{k} S(N,i)$ where $S(N,i)$ is the Stirling number of the second kind [28]

Similarity between time series and clusters :

A similarity criterion has been defined to assess the proximity between a new time series and the clusters of a coclustering model [29]. This criterion evaluates the decrease of the evaluation criterion *(Equation 1)* when a time series is merged within a particular cluster. Let us consider a coclustering model denoted by M, and M_{S_i,i_C} the same model where the time series S_i is merged within the cluster i_C. This criterion evaluates the difference $\Delta(S_i, i_C) = c(M_{S_i,i_C}) - c(M)$. Intuitively, if the curve S_i and the cluster i_C have a similar joint density $P(T,X)$, the total code length of the data *(see criterion $c(M)$)* is not much different when the curve is merged within its most similar cluster.

The computation of $\Delta(S_i, i_C)$ only implies the non-empty cells of the *ith* cluster of time series. The minimum value of Δ over all clusters is found by considering all non-empty cells of M. The number of non-empty cells is bounded by m. Consequently, the closest cluster of a time series is found with a $\mathcal{O}(m)$ time complexity. As explained in the next section, the criterion Δ is exploited by our approach in order to map sub-series with typical distributions.

C. Learning and deployment SAXO algorithms

SAXO includes two successive processes : i) the **learning** step consists in learning the symbolic representation from a sampled dataset; ii) the **deployment** step consists in converting the original large dataset into symbolic time series.

*/*Computation of time intervals*/*

(1) $\mathcal{D}_1 = Sample(\mathcal{D}, \delta)$
(2) $\mathcal{P}_1 = Encode(\mathcal{D}_1, \{C, T, X\})$
(3) $\{t\} \leftarrow Coclustering(\mathcal{P}_1, \{C, T, X\})$

*/*Computation of typical distributions within each interval*/*

For j **from** 1 **to** $|\{t\}|$ **do**

(A) $\mathcal{D}_2 = TimeFilter(\mathcal{D}_1, t_j)$
(B) $\mathcal{P}_2 = Encode(\mathcal{D}_2, \{C, X\})$
(C) $\{i_c\}^j \leftarrow Coclustering(\mathcal{P}_2, \{C, X\})$

end For

Algorithm 1: SAXO learning algorithm

Algorithm 1 describes the learning step. First of all, the dataset $\mathcal{D}$ is uniformly sampled according to a sampling rate δ *(Algorithm 1, line 1)*. The objective is to control the computing time of the algorithm. The times series of the resulting subset $\mathcal{D}_1$ are recoded by a set of data points

denoted by $\mathcal{P}_1$ *(line 2)*. As explained in Section III-B, each data point is characterized by three variables : C, T and X. A first trivariate coclustering model is trained on $\mathcal{P}_1$ by exploiting the MODL approach. Only the *time* discretization is retained *(line 3)*. The associated set of time intervals is denoted by $\{t\}$, and $|\{t\}|$ corresponds to the length of the SAXO words.

Notation for bivariate coclustering

- j *: index of the current time interval;*
- S_n^j *: sub-series computed from* S_n *within interval* j*;*
- $\{i_c\}^j$ *: set of clusters of sub-series within interval* j *;*
- k_{C_j} *: number of cluster of sub-series;*
- k_{X_j} *: number of intervals of values;*

For each time interval, the sampled dataset is filtered on the time period corresponding to the current interval *(line A)*. As previously, the outcome sub-series $\{S_n^j\}$ of $\mathcal{D}_2$ are recoded by a set of data points $\mathcal{P}_2$ *(line B)*. In this case, the data points are described by only two variables : C and X. Then, a bivariate coclustering model is trained on $\mathcal{P}_2$ *(line C)*. The clusters $\{i_c\}^j$ correspond to a set of typical distributions $P(X|C)$ approximated by a piecewise constant estimator. More precisely, $\{i_c\}^j$ denotes a set of distributions associated with the *jth* symbol of the SAXO representation.

The meaning of the symbols changes according to their rank because $P(X, C)$ is independently estimated over the time intervals. In addition, the size of the alphabet is not constant over intervals, because k_C is not necessary the same for all optimal bivariate coclustering models.

*/*Scan of the dataset $\mathcal{D}$*/*

For n **from** 1 **to** $|\mathcal{D}|$ **do**

  */*Initialization of the nth re-encoded time series*/*
  (A) $SAXO^n = \emptyset$

  */*For each time interval*/*
  For j **from** 1 **to** $|t|$ **do**
    (B) $S_n^j = TimeFilter(S_n, t_j)$
    (C) $Min\Delta = +\infty$
    (D) $id = 0$

    */*For each typical distribution*/*
    For i **from** 1 **to** $|\{i_c\}^j|$ **do**
      If $\Delta(S_n^j, i_{c_j}) < Min\Delta$ **then**
        (E) $id = i$
        (F) $Min\Delta = \Delta(S_n^j, i_{c_j})$
      end If
    end For
    (G) $SAXO^n = Concat(SAXO^n, Alpha^{id})$

  end For
end For

Algorithm 2: SAXO deployment algorithm

Algorithm 2 describes the deployment step of the SAXO approach. All time series $S_n \in \mathcal{D}$ are recoded by exploiting the results of Algorithm 1. The SAXO words are initialized as empty strings *(Algorithm 2, line A)*. Then a loop on the time intervals $\{t\}$ computes each symbol independently. The sub-series S_n^j belonging to the current time interval is filtered from S_n *(line B)*. Then, all clusters $\{i_c\}^j$ are evaluated in order to find the closest typical distribution which minimizes $\Delta(S_n^j, i_{c_j})$ *(defined in Section III-B)*. The minimal value of the similarity Δ is stored in the variable "$Min\Delta$" *(line C)* and the identifier of the closest cluster is stored in "id" *(line D)*. At last, the current SAXO word is updated : the *id-th* symbol of the alphabet $\{Alpha\}$ is concatenated at the end of the current word *(line G)*.

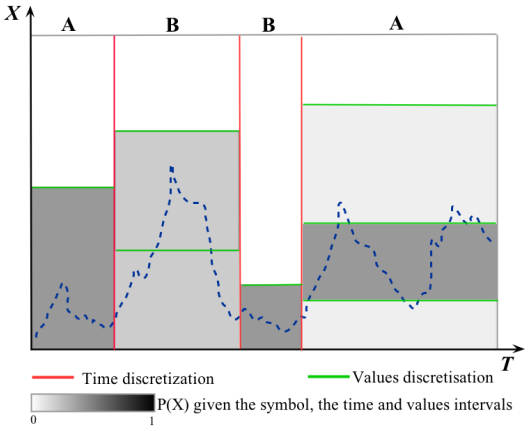

Fig. 5. Example of a SAXO representation.

Figure 5 plots an example of recoded time series. The original time series *(represented by the blue curve)* is recoded by the **"abba"** SAXO word. The time is discretized into four intervals *(the vertical red lines)* corresponding to each symbol. Within time intervals, the values are discretized *(the horizontal green lines)* : the number of intervals of values and their locations are not necessary the same. The symbols correspond to typical distributions of values : conditional probabilities of X are associated with each cell of the grid *(represented by the gray levels)*;

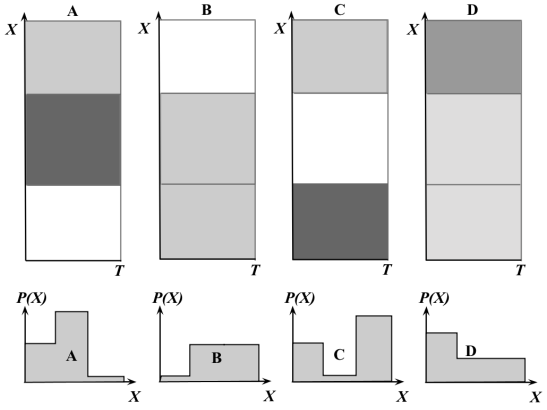

Fig. 6. Example of the alphabet of the second time interval.

Figure 6 gives an example of the alphabet associated with the second time interval. The four available symbols correspond to typical distributions which are both represented by gray levels and by histograms. By considering Figures 5 and 6, **b** appears to be the closest typical distribution of the second sub-series.

In Algorithms 1 and 2, the steps which require the largest computing time are respectively the coclustering and the similarity Δ. Based on this statement, the Algorithm 1 has a $\mathcal{O}(m\sqrt{m}\log m)$ time complexity and the Algorithm 2 has a $\mathcal{O}(m)$ complexity per serie to deploy *(see Section III-B)*. The learning phase is conducted by exploiting a sampled dataset, that allows us to process large datasets.

IV. Comparative experiments on supervised learning tasks

A. Experimental protocol

In this section, several symbolic representations of time series are evaluated conditionally to a supervised learning task. Our evaluation protocol is inspired by previous works which combine supervised and unsupervised approaches in order to evaluate the quality of the unsupervised task. For instance, the cascade evaluation [30] consists in enriching a supervised dataset with the *cluster id* of each example. Then, the *cluster id* is exploited by a classifier as an additional explicative variable. The cascade evaluation estimates the quality of the unsupervised task by measuring the improvement of the classifier when the *cluster id* is used.

In our experiments, the input dataset describes a collection of N labeled time series. The m_* numerical explicative variables correspond to the data points of time series[3]. The target variable encodes one of the Y possible class values, associated with each time series. Our comparative experiments aim to estimate the ability of symbolic representations to preserve the information of the original time series.

The input dataset is sequentially recoded into multiple symbolic versions *(the class value is ignored)*. The SAXO approach is first applied *(Section III)*. Using a trivariate coclustering *(Step 3 of Algorithm 1)*, the numbers of intervals k_T, k_X and the number of clusters k_C are automatically adjusted[4]. Then, the number of intervals k_{X_j} and the number of clusters k_{C_j} are also automatically adjusted by a bivariate coclustering, within each time interval[5] *(Step C of Algorithm 1)*. The SAX approach is applied *(Section II-B)* by exploiting the trivariate coclustering results to set its parameters *($w = k_T$ and $\alpha = k_X$)*. At last, the *Hybrid* approach exploits the *time* and *value* discretization resulting from the trivariate coclustering *(i.e. the numbers of intervals k_T, k_X, and the bounds location given by $\{m_{j_T}\}$ and $\{m_{j_X}\}$)*. Similarly to SAX, the *Hybrid* approach splits the input time series into k_T intervals while keeping the optimized bounds of the trivariate coclustering, and symbolizes the sub-series by recoding the mean values by the index of the X interval where it falls.

A naive Bayes classifier is trained on each version of the recoded dataset. In all cases, the symbols are exploited as categorical explicative variables without taking advantage of the meaning of symbols. Such a **weak classifier** allows us to evaluate the quality of the representations by **reducing the bias** due to an efficient learning algorithm. As a baseline approach, two naive Bayes classifiers are also trained on the original dataset : the m_* continuous explicative variables are discretized into 10 equal-frequency intervals and 10 equal-length intervals. In all cases, the classifiers are evaluated by using the multi-class AUC *(Area Under the ROC Curve)* [31].

The processed datasets come from the *UCR Time Series Classification and Clustering* repository [32]. Some datasets are relatively small, we have selected the ones which include at least 800 learning examples. Each dataset is split into a training set *(70%)* and a test set *(30%)* : these subsets are disjoint and randomly constituted. In our experiments, the relatively small size of the datasets does not require a sampling to learn the SAXO representation in an efficient way *($\delta = 1$ in Algorithm 1)*.

B. Comparative results

Table I shows the AUC reached by the naive Bayes classifier, considering the various representations of the time series. A color code gives the rank of each symbolic representations. Table I provides additional information related to the datasets such as the number of time series N, the number of target values Y, the length of the time series m_*. The parameters w and α of the SAX representation are also provided.

In all datasets except three *(AllFace, Faces UCR, TwoLeadECG)*, the naive Bayes classifier reaches higher performance by exploiting symbolic representations rather than the original time series. This result must be interpreted in light of the curse of dimensionality [7] which recommends few informative explicative variables. Table I can be interpreted in terms of compression by comparing the values of m_* and w, while keeping in mind that a *float* is coded by 32 bits and a *character* by 8 bits. **The average compression rate equals to** 95%.

According to J. Demšar, the *Wilcoxon signed-ranks test* must be exploited to reliably compare two classifiers over multiple datasets [33]. If the output value *(denoted by z)* is smaller than -1.96, the two classifiers have significantly different performance. The SAX and the hybrid representations can not be distinguished : in this case the Wilcoxon test gives $z = -0.365$. This surprising result reveals that **the optimization of the *time* and *values* discretization is not enought to improve the symbolic representation**. The SAXO approach reaches the best AUC in 83% of cases. The Wilcoxon test gives $z = -2.71$ when SAXO is compared with the SAX representation. The gap in performance between both representations is highly significant, **which demonstrates the interest of representing typical distributions by symbols** in addition to optimizing the *time* discretization.

[3]The N time series of the input dataset have the same length m_*.

[4]For a better interpretability of symbolic representations, we use at most 26 symbols in the alphabet, which corresponds to a maximum value of k_X.

[5]For comparison purpose, we impose our method to have an alphabet with no more than k_X symbols ($k_{C_j} \leq k_X \quad \forall j \in [1, k_T]$).

Type of Naive Bayes	Naive Grouping			10 Eq Freq	10 Eq Width	Additional Information				
Data representation	SAXO	SAX	Hybrid	Baseline: Original time series		N	Y	m_*	w	α
Starlightcurves	0.971	0.963	0.947	0.872	0.872	9 240	3	1024	46	26
AllFace	0.975	0.970	0.970	0.967	0.978	2 250	9	131	58	12
Mallat	0.998	0.999	0.997	0.996	0.997	2 400	8	1024	271	26
MoteStrain	0.982	0.960	0.959	0.951	0.947	1 270	2	84	15	16
Wafer	0.981	0.977	0.662	0.860	0.873	7 160	2	150	7	26
Yoga	0.875	0.834	0.840	0.786	0.781	3 300	2	426	69	21
uWaveGestureLib X	0.930	0.909	0.911	0.863	0.873	4 470	8	315	22	23
uWaveGestureLib Y	0.920	0.906	0.500	0.820	0.823	4 470	8	315	4	26
uWaveGestureLib Z	0.921	0.904	0.907	0.831	0.833	4 470	8	315	22	23
Cricket X	0.900	0.784	0.801	0.682	0.692	780	9	300	14	20
Cricket Y	0.915	0.797	0.836	0.748	0.771	780	9	300	14	26
Cricket Z	0.864	0.769	0.806	0.698	0.709	780	9	300	15	19
ECG Five Days	0.972	0.937	0.965	0.955	0.956	880	2	136	26	14
Faces UCR	0.979	0.975	0.975	0.969	0.981	2 250	9	131	63	13
Symbols	0.999	0.990	0.993	0.963	0.966	1 020	6	398	37	24
TwoLeadECG	0.888	0.915	0.906	0.929	0.955	1 160	2	82	30	19
50 Words	0.859	0.816	0.817	0.714	0.749	905	50	270	30	14
CBF	0.998	0.999	0.998	0.971	0.973	930	3	128	14	13
Chlorine Concentration	0.569	0.710	0.720	0.648	0.668	4 307	3	166	155	26
CinC EGC torso	0.999	0.994	0.998	0.980	0.988	1 420	4	1 639	47	26
Medical Images	0.837	0.796	0.824	0.766	0.824	1 041	10	99	15	20
SwedishLeaf	0.984	0.969	0.969	0.502	0.502	1 125	15	128	44	16
WordSynonyms	0.804	0.789	0.792	0.765	0.795	905	25	270	27	14

TABLE I

COMPARATIVE RESULTS (N IS THE NUMBER OF TIME SERIES IN THE DATASETS, Y IS THE NUMBER OF TARGET VALUES, m_* IS THE LENGTH OF THE ORIGINAL TIME SERIES , w IS THE NUMBER OF TIME INTERVALS, AND α IS THE MAX SIZE OF THE ALPHABET).

V. CONCLUSION AND PERSPECTIVES

This article introduces SAXO which is a new data-driven symbolic representation of time series. The main innovation of our approach is to represent typical distributions of data points by symbols, instead of average values. The definition of these typical distributions and the *time* discretization are optimized by a nonparametric Bayesian coclustering approach named MODL.

Our approach is divided into two successive steps : i) a learning algorithm defines the SAXO representation from a sampled dataset; ii) a deployment algorithm applies the SAXO representation on the overall dataset. The time complexities of both algorithms are consistent with the processing of large datasets.

Our approach is favorably compared with the SAX representation, by evaluating a classifier which is trained from recoded datasets. Our experiments highlight a significant gap in performance between both approaches, by using the *Wilcoxon signed-ranks* test applied on a collection of real datasets.

The SAXO approach has been successfully applied on large proprietary datasets, such as individual electricity consumptions. Ultimatelty, this approach may allow EDF to efficiently store, query and process large datasets of time series.

A similarity between two SAXO words could be defined and tested in future works. This way, SAXO would be exploited by similarity based learning algorithms *(ex : k-means, knn)* while keeping the advantage of considering distributions instead of numerical values. For instance, the possible application areas of such a similarity could be : i) the detection of atypical time series; ii) the query of a database by similarity; iii) the clustering of time series ... etc.

At last, an online version of the SAXO representation could be implemented in order to process data streams. In the near future, EDF will need to manage a large amount of daily updated metering data. In this context, an incremental update of the SAXO representation could be helpful to manage large and evolving datasets.

REFERENCES

[1] A. Antoniadis, X. Brossat, J. Cugliari, and J. M. Poggi. Clustering functional data using wavelets. In *COMPSTAT (International Conference on Computational Statistics)*, 2010.

[2] L. M. Sangalli, P. Secchi, S. Vantini, and V. Vitelli. K-mean Alignment for Curve Clustering. *Computational Statistics & Data Analysis*, 54(5):1219–1233, 2010.

[3] A. Ba, Y. Goude, M Sinn, and P. Pompey. Adaptive Learning of Smoothing Functions: Application to Electricity Load Forecasting. In *NIPS (Neural Information Processing Systems)*, 2012.

[4] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, San Diego, 2003.

[5] D. Bosq. *Linear Processes in Function Spaces: Theory and Applications (Lecture Notes in Statistics)*. Springer, 2000.

[6] J.O. Ramsay and B.W. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer, 2005.

[7] R.E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, USA, 1961.

[8] M. Frigo and S.G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[9] R. Polikar. *Physics and Modern Topics in Mechanical and Electrical Engineering*, chapter The story of wavelets. World Scientific and Eng. Society Press, 1999.

[10] K. Chan and W. Fu. Efficient Time Series Matching by Wavelets. In *ICDE '99: Proceedings of the 15th International Conference on Data Engineering*. IEEE Computer Society, 1999.

[11] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transfom. *IEEE Trans. Comput.*, 23(1):90–93, 1974.

[12] C. Guo, H. Li, and D. Pan. An improved piecewise aggregate approximation based on statistical features for time series mining. In *Proceedings of the 4th international conference on Knowledge science, engineering and management*, KSEM'10, pages 234–244, Berlin, Heidelberg, 2010. Springer-Verlag.

[13] A. Gionis and H. Mannila. Finding Recurrent Sources in Sequences. In *7th International Conference on Research in Computational Molecular Biology*, Berlin, 2003.

[14] G. Reinert, S. Schbath, and M.S. Waterman. Probabilistic and Statistical Properties of Words: An Overview. *Journal of Computational Biology*, 7:1–46, 2002.

[15] C.S. Daw, C.E.A. Finney, and E.R. Tracy. Symbolic Analysis of Experimental Data. *Review of Scientific Instruments*, 2002.

[16] C. Ratanamahatana, E. Keogh, T. Bagnall, and S. Lonardi. A Novel Bit Level Time Series Representation with Implications for Similarity Search and Clustering. In *PAKDD*, 2005.

[17] C. Cortes, K. Fisher, D. Pregibon, A. Rogers, and F. Smith. Hancock: a Language for Extracting Signatures from Data Streams. In *6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 9–17, Boston, 2000.

[18] E. Keogh, J. Lin, and A. Fu. HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In *5th IEEE International Conference on Data Mining*, pages 226–233, Houston, Texas, 2005.

[19] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic Discovery of Time Series Motifs. In *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 493–498, Washington, 2003.

[20] E. Keogh, L. Wei, X. Xi, S. Lonardi, J. Shieh, and S. Sirowy. Intelligent Icons: Integrating Lite-Weight Data Mining and Visualization into GUI Operating Systems. In *International Conference on Data Mining*, 2006.

[21] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh. iSAX 2.0: Indexing and Mining One Billion Time Series. In *International Conference on Data Mining*, 2010.

[22] E. Keogh, K. Chakraborty, S. Mehrotra, and M. Pazzani. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Journal of Knowledge and Information Systems*, 3:263–286, 2000.

[23] B. K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *26th International Conference on Very Large Databases*, Cairo, 2000.

[24] E. Keogh and S. Kasetty. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 102–111, Alberta, Canada, 2002.

[25] M. Boullé. *Hands on pattern recognition*, chapter Data grid models for preparation and modeling in supervised learning. Microtome, 2010.

[26] C.P. Robert. *The Bayesian Choice : From Decision Theoretic Foundations to Computational Implementation*. Springer, 2007.

[27] M. Boullé. Functional data clustering via piecewise constant nonparametric density estimation. *Pattern Recognition*, 45(12):4389–4401, 2012.

[28] M. Abramowitz and I.A Stegun. *Handbook of mathematical functions*. Dover, New York, 1965.

[29] R. Romain Guigourès, M Boullé, and F. Rossi. A Triclustering Approach for Time Evolving Graphs. In *ICDM Workshops*, 2012.

[30] L. Candillier, I. Tellier, F. Torre, and O. Bousquet. Cascade evaluation of clustering algorithms. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *17th European Conference on Machine Learning (ECML'2006)*, volume LNAI 4212 of *LNCS*, pages 574–581, Berlin, Germany, september 2006. Springer Verlag.

[31] T. Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. T. Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. Technical Report HPL-2003-4, HP Labs, 2003., 2003.

[32] E. Keogh, Q. Zhu, B. Hu, Hao. Y., X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage : www.cs.ucr.edu/∼eamonn/time_series_data/, 2011.

[33] J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, December 2006.