Enhancing Vector Comparison Using HMMs

Emmanuel Deli Madiga, Sylvain Iloga

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 August 2023, accepted 30 August 2023, date of publication 4 September 2023, date of current version 12 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3312019

RESEARCH ARTICLE

Enhancing Vector Comparison Using HMMs

EMMANUEL DELI MADIGA[1] AND SYLVAIN ILOGA[2,3,4,5]

[1]Department of Mathematics and Computer Science, Faculty of Sciences, The University of Maroua, Maroua, Cameroon
[2]Department of Computer Science, Higher Teachers' Training College, The University of Maroua, Maroua, Cameroon
[3]ENSEA, CNRS, ETIS UMR 8051, CY Cergy Paris University, 95000 Cergy, France
[4]ESIEE-IT, CCI Paris île-de-France, 95300 Pontoise, France
[5] CIRD, UMMISCO, University of Sorbonne, 93143 Bondy, France

Corresponding author: Sylvain Iloga (sylvain.iloga@gmail.com)

This work was supported by the Equipes Traitement de l'Information et Systèmes (ETIS) Unité mixte de recherche (UMR) 8051 Laboratory.

ABSTRACT Vectors are massively used in many domains. Several techniques have been proposed for comparing two vectors, but they only perform the comparison according to the exact values of the vector components. Additionally, existing techniques used for comparing two vectors having different dimensions are limited by many factors. Furthermore, the problem of comparing two finite sets of vectors has not yet been specifically addressed. This paper attempts to overcome all these limitations by proposing a new technique based on hidden Markov models which enhances existing techniques by giving them the ability to compare two finite sets of vectors, each containing vectors having different dimensions, while precising the set of targeted properties on which the comparison should be performed. Classification experiments conducted on three online available custom datasets demonstrated that when the suitable set of targeted properties is selected, the proposed approach outperforms existing techniques with accuracy gains reaching $+82.3\%$.

INDEX TERMS Vectors, vector comparison, distance between vectors, hidden Markov models.

I. INTRODUCTION

Vector-based descriptors are massively used in many domains including mathematics, physics, computer science, etc. For this reason, several multidisciplinary repositories containing datasets where data are represented as vectors are available. This is the case of the widespread *UCI Machine Learning Repository*[1] which contains more than 550 datasets from various domains. A vector $\overrightarrow{u} = [u_1, u_2, \ldots, u_n]$ is basically an ordered list of n elements belonging to diverse datatypes. In this paper, the vector components belong to $\mathbb{N}$ or to $\mathbb{R}$. The number n of components of $\overrightarrow{u}$ is its *dimension*.

Vectors can be manipulated by different operations including vector comparison which is crucial for popular machine learning tasks like *classification* or *clustering*. Given a dataset composed of many classes, each class containing several vectors, classification refers to the process of assigning a class label to a vector based on its components [1]. Clustering is a method for organizing vectors into groups with the most similarity in the same cluster and the most dissimilarity between different clusters [2].

Depending on the final goal of the comparison (decision support, classification, clustering, etc.), several **distance** [3] and **similarity** [4] measures have already been proposed. These measures are generally suitable for comparing two single vectors having the same dimension, according to the exact values of their components. These existing measures do not enable to explicitly specify the criteria on which the comparison must be performed. However, the result $d(u, v)$ of the comparison of two items u and v represented by vector data is not unique, but rather depends on the set P of comparison criteria. For example, consider the two cars u and v respectively depicted in Figures 1a and 1b. When $P = \{numberofdoors, carcolor\}$, the comparison result is $d(u, v) = 'similar'$. However, the opposite result $d(u, v) = 'not\ similar'$ is obtained for these same cars when $P = \{enginepower, numberofwheels, carweight\}$. Although these two comparison results are different, they both remain fully consistent according to their corresponding sets of comparison criteria. More generally, the construction of P must be guided by the final goal of the comparison. But whatever is the content of P, the comparison result remains consistent according to P.

In the same way, the comparison of two numeric vectors can be performed according to many other targeted properties

The associate editor coordinating the review of this manuscript and approving it for publication was Yun Lin.

[1]https://archive.ics.uci.edu/ml/

(a) Car u (b) Car v

FIGURE 1. Example of cars compared according different sets of criteria.

than the exact values of their components. As an example, the vectors $\overrightarrow{u} = [44, 156, 84, 548]$ and $\overrightarrow{v} = [188, 4, 326, 142]$ are very distant according to the exact values of their components, but these vectors become very close when the *parity* of these same components is considered.

Additionally, when arises the problem of comparing two vectors having different dimensions, existing measures are unsuitable. A typical scenario in real-world demonstrating the need of comparing two vectors having different dimensions can be described as follows: Consider two people u and v, each being the owner of a unique bank account. Suppose now that we want to compare u and v according to the supplies and withdrawals of cash they perform in their respective bank accounts over a fixed period (one year for example). To achieve this, we decide to record the actions (supply/withdrawal) of u in a vector $\overrightarrow{u}$ and those of v in a vector $\overrightarrow{v}$ such that the k^{th} component of $\overrightarrow{u}$ (resp. $\overrightarrow{v}$) contains the k^{th} action performed by $\overrightarrow{u}$ (resp. $\overrightarrow{v}$), irrespective of the exact moment (date, time) when the action is performed. A supply of an amount a is recorded positively as $+a$, while a withdrawal of an amount a is recorded negatively as $-a$. If at the end of the fixed period u performs n actions and v performs m actions, this leads to the comparison of two vectors $\overrightarrow{u}$ and $\overrightarrow{v}$ of different dimensions.

Consider two vectors $\overrightarrow{u} \in \mathbb{R}^n$ and $\overrightarrow{v} \in \mathbb{R}^m$ such that $(n > m)$. Existing methods for comparing $\overrightarrow{u}$ and $\overrightarrow{v}$ most often consist either in applying *zero-padding* (ZP) techniques [5] to augment the dimension of $\overrightarrow{v}$ by adding zeros to each missing dimension or *dimensionality reduction* (DR) techniques [6], [7] for reducing the dimension of $\overrightarrow{u}$ such that both vectors finally have identical dimensions. But proceeding this way systematically implies a pure distortion of the reality, which may introduce an important bias in the final interpretation of the comparison result. Indeed, considering the former example, adding zeros is considered as the creation of supplies/withdrawals of zero which never occurred in the reality. For this same example, reducing the dimension of a vector by using *DR* techniques implies both, the deletion and the modification of actions which effectively occurred in the reality. This is because the components of a vector resulting from the application of *DR* techniques generally differ from those of its corresponding original vector. Consequently, an account supply may become a withdrawal and vice versa, or the final amount of a supply/withdrawal may become different from the corresponding original amount.

An interesting alternative for comparing two vectors $\overrightarrow{u}$ and $\overrightarrow{v}$ having different dimensions without distorting the reality is offered by the use of *Siamese neural networks* (SNN) [8]. These are two artificial neural networks, working in tandem and which achieve this task through a parallel learning process. One of the two neural networks produces as output a feature vector $f(\overrightarrow{u})$ associated with $\overrightarrow{u}$, while the other neural network produces as output a feature vector $g(\overrightarrow{v})$ associated with $\overrightarrow{v}$. The two outputted feature vectors have the same number of components. The final step consists in comparing $f(\overrightarrow{u})$ and $g(\overrightarrow{v})$ using existing measures, generally the *Cosine similarity* is preferred.[2] Unfortunately, this solution inherits the following drawbacks generally embedded by deep learning techniques analyzed in [9][3]:

1) They require lots of computing resources.
2) The parameters of the resulting models are difficult to adjust.
3) The components of the resulting feature vectors are less interpretable.

Furthermore, all existing techniques for comparing vectors do not explicitly address the problem of comparing two finite sets of vectors. Existing distance measures between two clusters attempt to solve this problem [10], but they do not consider the individual properties of the vectors in each set (clusters).

This paper attempts to overcome all the aforementioned limitations of existing work through the proposal of a new technique based on hidden Markov models (HMM) which enhances existing techniques by giving them the ability to compare two finite sets of vectors, each set containing vectors having different dimensions, while precising the set of targeted properties on which the comparison should be performed. The performance of the proposed technique is finally evaluated through flat classification experiments involving three online available custom databases especially designed for this purpose.

The rest of this paper is organized as follows: the state of the art is presented in Section II, followed by a synthetic description of HMM in Section III. The proposed approach is described in Section IV, while experimental results are presented in Section V. The last section is devoted to the conclusion.

II. STATE OF THE ART

A. EXISTING DISTANCES

In order to compare two vectors $\overrightarrow{u}, \overrightarrow{v} \in \mathbb{R}^n$, many distances and similarities have already been proposed. A distance is a function d which outputs a real number $d(\overrightarrow{u}, \overrightarrow{v})$ satisfying the four following axioms [11][4]:

1) $d(\overrightarrow{u}, \overrightarrow{v}) \geq 0$
2) $d(\overrightarrow{u}, \overrightarrow{v}) = 0 \Leftrightarrow (\overrightarrow{u} = \overrightarrow{v})$
3) $d(\overrightarrow{u}, \overrightarrow{v}) = d(\overrightarrow{v}, \overrightarrow{u})$

[2] See Figure 1 of [8] for details.
[3] See the Introduction of [9].
[4] See Definition 1 of [11].

4) $d(\overrightarrow{u}, \overrightarrow{w}) + d(\overrightarrow{w}, \overrightarrow{v}) \geq d(\overrightarrow{u}, \overrightarrow{v}), \forall \overrightarrow{w} \in \mathbb{R}^n$

For a purely statistical comparison, the most used distance for comparing two vectors of $\mathbb{R}^n$ is the *Euclidean distance* which evaluates the straight-line distance between them. Another popular distance is the *Manhattan distance* which evaluates the distance traveled by a taxi when it moves from one vector to the other in a city where the streets are organized as a grid. One can also use the *Chebyshev distance* to evaluate the maximum component-to-component variation between the two vectors. The *Squared* χ^2 *distance* is suitable for evaluating the dependency between two vectors and the *Canberra distance* is a weighted version of the *Manhattan distance*. The *Earth Mover's distance* offers a good alternative for comparing two vectors which are probability distributions by evaluating the least amount of work needed to transport earth or mass from one distribution to the other [12]. Formal details about more than 60 distance measures (including those listed in this section) are available in [3][5] and Equation 1 shows how to compute the aforementioned distances for two vectors $\overrightarrow{u} = [u_1, \ldots, u_n]$ and $\overrightarrow{v} = [v_1, \ldots, v_n]$.

$$
\begin{aligned}
d_1(\overrightarrow{u}, \overrightarrow{v}) &= \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2} && (Euclidean)\\
d_2(\overrightarrow{u}, \overrightarrow{v}) &= \sum_{i=1}^{n} |u_i - v_i| && (Manhattan)\\
d_3(\overrightarrow{u}, \overrightarrow{v}) &= \max_{1 \leq i \leq n} |u_i - v_i| && (Chebyshev)\\
d_4(\overrightarrow{u}, \overrightarrow{v}) &= \sum_{i=1}^{n} \frac{(u_i - v_i)^2}{u_i + v_i} && (Squared\ \chi^2)\\
d_5(\overrightarrow{u}, \overrightarrow{v}) &= \sum_{i=1}^{n} \frac{|u_i - v_i|}{|u_i| + |v_i|} && (Canberra) \qquad (1)
\end{aligned}
$$

B. EXISTING SIMILARITIES

Given two vectors $\overrightarrow{u}, \overrightarrow{v} \in \mathbb{R}^n$, a similarity is a function d which also outputs a real number $d(\overrightarrow{u}, \overrightarrow{v})$ satisfying the five following axioms [11][6]:

1) $d(\overrightarrow{u}, \overrightarrow{v}) = d(\overrightarrow{v}, \overrightarrow{u})$
2) $d(\overrightarrow{u}, \overrightarrow{u}) \geq 0$
3) $d(\overrightarrow{u}, \overrightarrow{u}) \geq d(\overrightarrow{u}, \overrightarrow{v})$
4) $d(\overrightarrow{u}, \overrightarrow{v}) + d(\overrightarrow{v}, \overrightarrow{w}) \leq d(\overrightarrow{u}, \overrightarrow{w}) + d(\overrightarrow{v}, \overrightarrow{v}),$ $\forall \overrightarrow{w} \in \mathbb{R}^n$
5) $d(\overrightarrow{u}, \overrightarrow{u}) = d(\overrightarrow{v}, \overrightarrow{v}) = d(\overrightarrow{u}, \overrightarrow{v}) \Leftrightarrow (\overrightarrow{u} = \overrightarrow{v})$

Many similarity measures also enable to compare two vectors having the same dimension. This is the case of the *Cosine similarity* which is the cosine of the angle $\widehat{(\overrightarrow{u}, \overrightarrow{v})}$ between $\overrightarrow{u}$ and $\overrightarrow{v}$. The value of this angle is consequently obtained by computing the arccos of the *Cosine similarity* as shown in Equation 2. When the comparison aims at discovering any rank correspondence between $\overrightarrow{u}$ and $\overrightarrow{v}$, several correlation coefficients can be computed. Among the most popular correlation coefficients, we can list: the *Pearson correlation coefficient*, the *Spearman's* ρ *rank coefficient*, the *Kendall* τ *coefficient* and the *Goodman–Kruskal's* γ *rank correlation*. A detailed analysis of these correlation coefficients is available in [4].[7]

$$
\begin{aligned}
Cosine(\overrightarrow{u}, \overrightarrow{v}) &= \frac{\sum_{i=1}^{n} u_i \times v_i}{\left(\sqrt{\sum_{i=1}^{n} {u_i}^2}\right) \times \left(\sqrt{\sum_{i=1}^{n} {v_i}^2}\right)}\\
\widehat{(\overrightarrow{u}, \overrightarrow{v})} &= \arccos\left(Cosine(\overrightarrow{u}, \overrightarrow{v})\right) \qquad (2)
\end{aligned}
$$

C. ZERO-PADDING

ZP [5] is an obvious solution for comparing two vectors having different dimensions. This consists in augmenting the required number of zeros at the beginning/middle/end of the vector having the lower dimension such that the two vectors finally have the same dimension.[8] As an example, before comparing $\overrightarrow{u} = [3, -7]$ and $\overrightarrow{v} = [18, -4, -26, 14]$, the vector $\overrightarrow{u}$ can initially be zero-padded to obtain $\overrightarrow{u_1}' = [3, -7, \mathbf{0}, \mathbf{0}]$, or $\overrightarrow{u_2}' = [\mathbf{0}, \mathbf{0}, 3, -7]$ or $\overrightarrow{u_3}' = [3, \mathbf{0}, \mathbf{0}, -7]$ before performing the comparison.

D. DIMENSIONALITY REDUCTION

An alternative to the use of *ZP* techniques for comparing vectors having different dimensions, is the use of *DR* techniques. *DR* techniques are thoroughly used in several application domains including face/image/pattern recognition, image/video classification and clustering, signal processing, etc.[9] In these domains, *DR* techniques are generally used for extracting only relevant components while eliminating redundant and unnecessary components. Given a dataset $U = \{\overrightarrow{u_1}, \ldots, \overrightarrow{u_x}\}$ where $\overrightarrow{u_i} \in \mathbb{R}^n$ $(1 \leq i \leq x)$, *DR* techniques aim at transforming U into the set $U' = \{\overrightarrow{u_1}', \ldots, \overrightarrow{u_x}'\}$ such that $\overrightarrow{u_i}' \in \mathbb{R}^m$ $(1 \leq i \leq x)$ with $(m \ll n)$, while preserving as much as possible a property of the original vectors. This property can be the *distance*, the *variance*, the *classification accuracy*, etc.[10] Several *DR* techniques have yet been proposed and depending on the technique, one can have the implicit, explicit or inverse mapping possibility to reconstruct an original vector $\overrightarrow{u_i} \in \mathbb{R}^n$ from its low dimensional correspondent $\overrightarrow{u_i}' \in \mathbb{R}^m$ [13].

DR techniques are organized in many categories, each category having a specific objective and including many specific techniques. A total of 12 categories together with their objectives are presented in [6].[11] Among these existing categories, *Linear DR techniques* [14], [15], [16], [17], [18], [19], [20], [21] and *Non-linear DR techniques* [22], [23], [24], [25], [26], [27], [28], [29], [30] are the most popular. *Linear DR techniques* aim at representing each reduced dimension as a linear combination of the original

[5] See Tables 1 to 8 and Table 10 of [3].
[6] See Definition 2 of [11].
[7] See Section II-E of [4].
[8] See Equations 4 to 13 of [5].
[9] See Table 4 of [6].
[10] See Table 2 of [6].
[11] See Table 1 of [6].

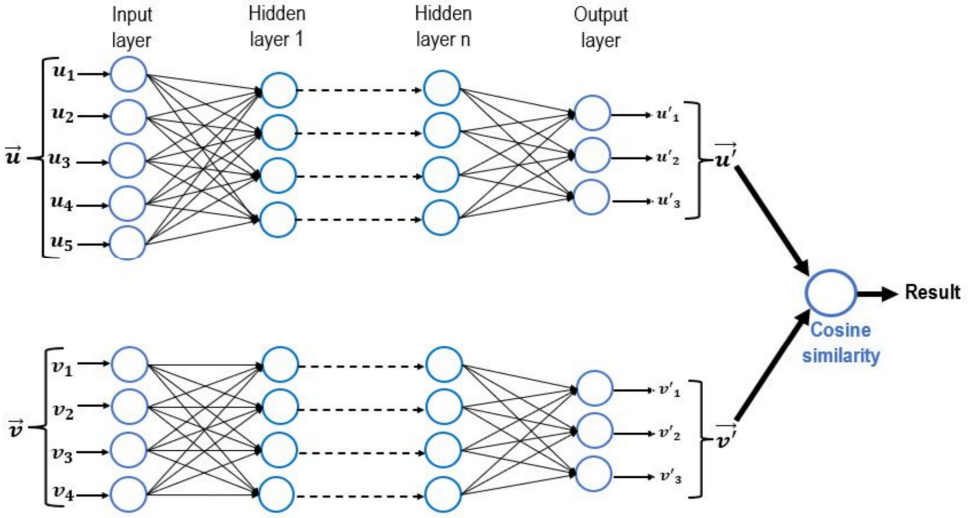

FIGURE 2. Example of vector comparison using SNN.

dimensions. *Principal Component Analysis* (PCA) [15], *Singular Value Decomposition* (SVD) [16], *Latent Semantic Analysis* (LSA) [17], *Locality Preserving Projections* (LPP) [18], *Independent Component Analysis* (ICA) [19], *Linear Discriminant Analysis* (LDA) [20] and *Projection Pursuit* (PP) [21] are members of this category. In *Nonlinear DR techniques*, each reduced dimension results from a nonlinear transformation of the original dimensions. This category includes the following specific techniques: *Kernel Principal Component Analysis* (KPCA) [23], *Multidimensional Scaling* (MS) [24], *Isomap* [25], [26], *Locally Linear Embedding* (LLE) [27], *Self-Organizing Map* (SOM) [28], *Learning Vector Quantization* (LVQ) [29] and *t-Stochastic Neighbor Embedding* (tSNE) [30].

All the aforementioned DR techniques are generally separately applied before to perform classification or clustering on a dataset. However, it is possible to directly perform the DR within the clustering algorithm as it is the case in [31] where a feature-reduction multi-view *k-Means* clustering algorithm is proposed. In that work, an objective function which automatically computes the weight associated with each feature is first designed. Irrelevant features with small weights are then eliminated in each view.

E. SIAMESE NEURAL NETWORKS

Another alternative for comparing two vectors having different dimensions consists in using *SNN*. A recent survey related to *SNN* and their applications is proposed in [8]. These models are generally used in audio/speech processing, image/video analysis, physics, robotics, text mining, etc.[12] *SNN* are two artificial neural networks working parallelly in tandem, each capable of generating one feature vector (called *hidden representation*) of each input vector through a machine learning process. The two neural networks are both feedforward perceptrons and perform error back-propagation during training. At the end of the training, these models usually compare their outputs through the computation of the *Cosine similarity* presented in Equation 2. As an example, Figure 2 depicts a *SNN* where the first artificial neural network takes as input the five components of a vector $\overrightarrow{u}$ and the second artificial neural network takes as input the four components of a vector $\overrightarrow{v}$. After the learning process, the system produces a three-dimensional feature vector for each input vector. The *Cosine distance* between the resulting feature vectors enables to obtain the final result. Table 1 summarizes the main information related to the existing techniques reviewed in this paper.

[12] See Section III of [8].

F. PROBLEM STATEMENT

Existing *distance* [3] and *similarity* [4] measures only target the exact values of the components of two vectors for comparing them. However, depending on the selected application, the comparison may target many other user-defined criteria. Existing measures are also unsuitable for comparing two vectors having different dimensions. *ZP techniques* [5] and *DR techniques* [6], [7], [31] which are most often used in that situation generally distort the reality through the creation of events that never occurred in the reality, or the deletion/modification of events that effectively occurred in the reality. *SNN* [8] offer an interesting alternative for avoiding the distortion of the reality. But, they are limited by several factors. Furthermore, the problem of comparing two finite sets of vectors has not yet been specifically addressed.

This paper attempts to overcome all these limitations by proposing a customizable technique based on HMM for comparing two finite sets of vectors, each set containing vectors having different dimensions. HMM are preferred here because they are suitable for learning sequential data and a vector can naturally be viewed as a sequential list of components. The ability of a HMM to learn multiple

TABLE 1. Main existing techniques for comparing two vectors.

General techniques	Surveyed by	Year	Specific techniques	Drawbacks
Distances	Cha [3]	2007	*Euclidean*	Limited to the comparison of two vectors having the same dimension.
			Manhattan	
			Chebyshev	
			Squared χ^2	
			Canberra	
	Yossi Rubner et al. [12]	2000	*Earth Mover's*	
Similarities	Zaid [4]	2015	*Cosine*	
			Pearson	
			Spearman's ρ	
			Kendall τ	
			Goodman–Kruskal's γ	
ZP	Al-Jawhar et al. [5]	2018	*ZP*	Distort the reality by creating, deleting or modifying events in the reality.
Linear *DR*	Cunningham et al. [14]	2015	*PCA* [15]	
			SVD [16]	
			LSA [17]	
			LPP [18]	
			ICA [19]	
			LDA [20]	
			PP [21]	
Non-linear *DR*	DeMers et al. [22]	1993	*KPCA* [23]	
			MS [24]	
			Isomap [25], [26]	
			LLE [27]	
			SOM [28]	
			LVQ [29]	
			tSNE [30]	
DR in clustering	Miin-Shen Yang et al. [31]	2019	Feature-reduction multi-view *k-Means*	
SNN	Chicco [8]	2021	*SNN*	- Lots of computing resources - Less interpretable outputs - Parameters difficult to adjust

sequences offers the additional possibility of associating one single HMM to a finite set of vectors. This later enables deriving one single feature vector for the considered set of vectors.

III. PRESENTATION OF HMM

A. DEFINITION OF A HMM

A HMM $\lambda = (A, B, \pi)$ is fully characterized by [32][13]:

1) Its number N of states, the set of states being $S = \{s_1, \ldots, s_N\}$. The state of the model at time t is generally noted $q_t \in S$.
2) Its number M of symbols, the set of symbols being $\vartheta = \{z_1, \ldots, z_M\}$. The symbol observed at time t is generally noted $o_t \in \vartheta$.
3) Its state transition probability matrix A verifying $A[s_i, s_j] = Prob(q_{t+1} = s_j | q_t = s_i)$ with $1 \leq i, j \leq N$.
4) Its symbols probabilities matrix B verifying $B[s_i, z_k] = Prob(z_k \text{ at time } t | q_t = s_i)$ with $1 \leq i \leq N$ and $1 \leq k \leq M$.
5) Its initial state probability vector π verifying $\pi[s_i] = Prob(q_1 = s_i)$ with $1 \leq i \leq N$.

B. GENERATION OF SEQUENCE

A HMM $\lambda = (A, B, \pi)$ can be used for generating a sequence $O = o_1 \ldots o_T$ composed of T symbols observed following the sequence of states $q = q_1 \ldots q_T$ as described in the *Markov chain* (MC) shown in Figure 3.

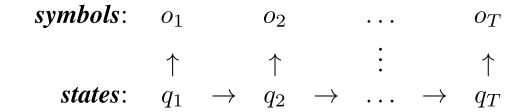

FIGURE 3. HMM used as sequence generator.

In order to obtain this MC, the following algorithm is executed:

1) Select the initial state s_j according to π and set $t = 0$.
2) Set $t = t + 1$ and change the current state to $q_t = s_j$
3) Select the symbol $o_t \in \vartheta$ to be observed at state q_t according to B.
4) If $(t < T)$ **go to** step 5, else **terminate**.
5) Select the state transition to be realized from the current state q_t to another state $s_j \in S$ according to A, then **go to** step 2.

C. MANIPULATION OF A HMM

Consider a sequence of symbols $O = o_1 \ldots o_T$ and a HMM $\lambda = (A, B, \pi)$. The probability $Prob(O|\lambda)$ to observe O given λ is efficiently calculated by the *Forward-Backward* algorithm [32][14] which runs in $\theta(T.N^2)$. Given a sequence of symbols $O = o_1 \ldots o_T$, it is possible to iteratively re-estimate

[13] See Section II-B of [32].

[14] See Section III-A of [32].

the parameters of an initial HMM $\hat{\lambda} = (\hat{A}, \hat{B}, \hat{\pi})$ in order to maximize the value of $Prob(O|\lambda)$, where $\lambda = (A, B, \pi)$ is the re-estimated model. The *Baum-Welch* algorithm [32][15] is used for this purpose. This algorithm runs in $\theta(\gamma.T.N^2)$ where γ is the user-defined maximum number of iterations.

The *Baum-Welch* algorithm can also be used to train a HMM for learning multiple sequences.[16] In this case, the algorithm maximizes the value of $Prob(O|\lambda) = \sum_{k=1}^{K} Prob(O^{(k)}|\lambda)$ where $O = \{O^{(1)}, \ldots, O^{(K)}\}$ is a set of K sequences and $O^{(k)} = O_1^{(k)} \ldots O_{T_k}^{(k)}$ is the k^{th} sequence of O. In this context, only the distributions A and B are re-estimated by the *Baum-Welch* algorithm because π can be statistically determined from the initial states of the K observed sequences. The time cost of the *Baum-Welch* algorithm for multiple sequences is approximated by $\theta\left(\gamma.(\sum_{k=1}^{K} T_k).N^2\right)$.

D. STATIONARY DISTRIBUTION

A vector $\varphi = (\varphi[s_1], \ldots, \varphi[s_N])$ is a stationary distribution of a HMM $\lambda = (A, B, \pi)$ if [33][17]:

1) $\forall j, \varphi[s_j] \geq 0$ and $\sum_j \varphi[s_j] = 1$
2) $\varphi = \varphi.A \Leftrightarrow \left(\varphi[s_j] = \sum_i \varphi[s_i] \times A[s_i, s_j], \forall j\right)$

$\varphi[s_j]$ estimates the overall proportion of time spent by λ in state s_j after a sufficiently long time. φ can be extracted from any line of the matrix $A^r = A \times A \times \ldots \times A$ (r times) when $r \to +\infty$. The computation of φ requires $\theta(r.N^3)$ arithmetic operations.

IV. THE PROPOSED APPROACH

A. MAIN IDEA

Consider a vector $\overrightarrow{u} = [u_1, \ldots, u_n]$ and let note $\overrightarrow{e_k}$ $(1 \leq k \leq n)$ the vector of $\mathbb{R}^n$ whose components are all equal to 0, except the k^{th} component which is equal to 1. In these conditions, $\overrightarrow{u} = \left(\sum_{k=1}^{n} u_k.\overrightarrow{e_k}\right)$. In $\mathbb{R}^3$ for example, $\overrightarrow{e_1} = [1, 0, 0]$, $\overrightarrow{e_2} = [0, 1, 0]$ and $\overrightarrow{e_3} = [0, 0, 1]$. Given a user-defined targeted property p, the main idea of the current work is that $\overrightarrow{u}$ can be characterized by the adherence of its components to p.

To achieve this goal, we behave like a photograph who sequentially browses $\overrightarrow{u}$ from its first dimension to its last dimension, and in each dimension k $(1 \leq k \leq n)$, the photograph takes a *suitable position* for capturing the *value of* $p(u_k)$ which materializes the adherence of the k^{th} component u_k to property p. In the current work, this suitable position is the angle $\widehat{(\overrightarrow{u}, \overrightarrow{e_k})}$ which is computed using Equation 2. Proceeding this way, the pseudo MC depicted in Figure 4 is obtained.

Any property p related to the k^{th} component u_k of a vector $\overrightarrow{u} = [u_1, \ldots, u_n]$ having $(n \geq 2)$ components can be targeted during vector comparison. Properties p_1 to p_5 listed below are examples of such properties:

1) $p_1(u_k) = u_k$

[15] See Section III-C of [32].
[16] See Section V-B of [32].
[17] See Definition 9.1 of [33].

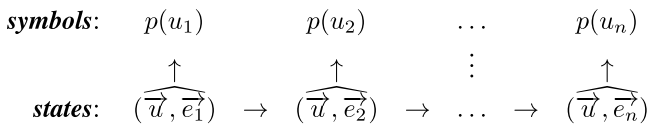

FIGURE 4. Pseudo MC associated with $\overrightarrow{u} = [u_1, \ldots, u_n]$ according to property p.

TABLE 2. Property p_6. The symbols e and o respectively stand for '*even*' and '*odd*'.

	$\lvert u_{k-1}\rvert$	$\lvert u_k\rvert$	$\lvert u_{k+1}\rvert$	$p_6(u_k)$
$k \in [2, n-1]$	o	o	o	1
	o	o	e	2
	e	o	o	3
	e	o	e	4
	e	e	e	5
	e	e	o	6
	o	e	e	7
	o	e	o	8
$k = 1$		o	o	9
		o	e	10
		e	e	11
		e	o	12
$k = n$	o	o		13
	e	o		14
	e	e		15
	o	e		16

2) $p_2(u_k) = 1$ if $\lvert u_k\rvert$ is even, 0 otherwise.
3) $p_3(u_k) = 1$ if $\lvert u_k\rvert$ is a first number, 0 otherwise.
4) $p_4(u_k) = (\lvert u_k\rvert \bmod 10)$
5) $p_5(u_k) = 100.\frac{|W|}{(n-1)}$ where $W = \{u_i | i \neq k, u_i \geq u_k\}$

Property p_1 returns the exact value of u_k, while property p_2 returns the parity of $\lvert u_k\rvert$. Property p_3 checks if $\lvert u_k\rvert$ is a first number and property p_4 returns the last digit of $\lvert u_k\rvert$ (when u_k is an integer). Finally, property p_5 returns the proportion of components of $\overrightarrow{u}$ which are greater or equal to u_k. The targeted property p may not only be related to the k^{th} component of $\overrightarrow{u}$, but also to its immediate eventual left/right neighbor(s). Properties p_6 and p_7 respectively presented in Tables 2 and 3 are examples of such properties. Properties p_2 and p_6 both target the parity of $\lvert u_k\rvert$, but p_6 is finer because it considers the neighbors. Similarly, properties p_3 and p_7 both aim at verifying if $\lvert u_k\rvert$ is a first number or not, but p_7 is finer because it consider the neighbors.

An important issue arising from Figure 4 is that the states are angles belonging to the continuous interval [0,180] (in degrees), however the set of states of a HMM must be finite. Another issue can occur when property p returns a real number, however the set of symbols of a HMM must also be finite. This is the case for properties p_1 and p_5. Solutions for these issues are proposed in Section IV-C.

B. METHODOLOGY

As it can be observed in Figure 5, traditional machine learning techniques for vector comparison generally take as inputs two vectors of different dimensions and learn the exact values of their components in order to extract two feature vectors having the same dimension, before to perform the comparison using existing measures. *SNN* are examples of existing

TABLE 3. Property p_7. The symbols + and − respectively stand for *'first'* and *'not first'*.

	$\lvert u_{k-1}\rvert$	$\lvert u_k\rvert$	$\lvert u_{k+1}\rvert$	$p_7(u_k)$
$k\in[2, n-1]$	−	−	−	1
	−	−	+	2
	+	−	−	3
	+	−	+	4
	+	+	+	5
	+	+	−	6
	−	+	+	7
	−	+	−	8
$k=1$		−	−	9
		−	+	10
		+	+	11
		+	−	12
$k=n$	−	−		13
	+	−		14
	+	+		15
	−	+		16

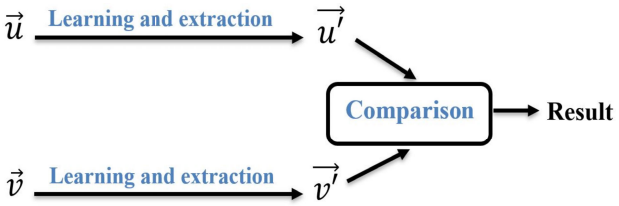

FIGURE 5. Traditional methodology for vector comparison using machine learning.

techniques following the traditional methodology presented in Figure 5. On the contrary, the methodology presented in Figure 6 shows that the proposed approach rather takes as inputs two finite sets of vectors having different dimensions and requires the additional specification of a set of targeted properties before to generate one feature vector for each input set. The final comparison result is also obtained by comparing the generated feature vectors using existing measures. More formally, given a user-defined finite set $P = \{p_1, \ldots, p_l\}$ of l targeted properties, the proposed methodology for generating the final feature vector $\overrightarrow{U_P}$ associated with a set of vectors $U = \{\overrightarrow{u_1}, \ldots, \overrightarrow{u_x}\}$ is composed of two main steps:

1) For each $p \in P$, generate the single feature vector U_p as described in Figure 7.
2) Generate the final feature vector $\overrightarrow{U_P}$ by concatenating all its corresponding single feature vectors as described in Figure 8.

For each $p \in P$, the generation of the single feature vector is realized as follows:

1) **Transformation into MC**: The adherence of the components of every $\overrightarrow{u_i} \in U$ $(1{\leq}i{\leq}x)$ to p is evaluated and saved into the MC $\delta_p(\overrightarrow{u_i})$.
2) **HMM initialization and training**: The resulting set $\{\delta_p(\overrightarrow{u_1}), \ldots, \delta_p(\overrightarrow{u_x})\}$ is used for initializing and training the HMM λ_p^U associated with U according to p.
3) **Meta-data extraction**: Meta-data extracted from λ_p^U are finally used for generating the single feature vector U_p associated with U according to p. These meta-data are related to the overall time spent by λ_p^U observing each symbol after a sufficiently long time.

C. TRANSFORMATION INTO MC

Given a vector $\overrightarrow{u} = [u_1, \ldots, u_n]$, a user-defined integer N and a user-defined property p, the MC $\delta_p(\overrightarrow{u})$ associated with $\overrightarrow{u}$ according to property p is constructed as described in Figure 4. In order to solve the first issue related to the fact that the states are angles belonging to the continuous interval [0,180] in Figure 4, we first split the interval [0,180] into $(N+1)$ slices $\{s_0, s_1, \ldots, s_N\}$ as defined in Equation 3 following [34][18] where the authors split the interval [0, 100] into $(N+1)$ slices using a similar technique.

$$\begin{cases} s_0 = \{0\} \\ s_j = \left]\dfrac{180}{N}\times(j-1), \dfrac{180}{N}\times j\right], \ (1{\leq}j{\leq}N) \end{cases} \tag{3}$$

If the value of N used for splitting the interval [0, 180] is high, then the length of each slice s_j becomes tiny and all the elements in s_j become very near to one unique value which is $\frac{180}{N}\times j$. The elements of s_j can therefore be considered as one single element that we identify here by the index j of the slice s_j. This reasoning allows us to derive the MC $\delta_p(\overrightarrow{u})$ associated with $\overrightarrow{u}$ according to property p by replacing each angle appearing in Figure 4 by the index j of the slice s_j to which it belongs. When this principle is applied to the content of a set $U = \{\overrightarrow{u_1}, \ldots, \overrightarrow{u_x}\}$ of vectors, each $\overrightarrow{u_i}$ $(1{\leq}i{\leq}x)$ having its specific dimension, the set $\Delta_p^U = \{\delta_p(\overrightarrow{u_1}), \ldots, \delta_p(\overrightarrow{u_x})\}$ of MC is obtained.

As an example, consider the singleton $U = \{\overrightarrow{u}\}$ where $\overrightarrow{u} = [17, 47, 81, 20]$ and suppose that $N = 10$ for splitting the interval $[0, 180]$ into the 11 following slices: $s_0 = \{0\}$, $s_1 =]0, 18]$, $s_2 =]18, 36]$, …, $s_{10} =]162, 180]$. In these conditions, if properties p_4, p_6 and p_7 are individually selected, $\overrightarrow{u}$ is first transformed into the pseudo MC presented in Figures 9a, 9c and 9e respectively, before deriving its final MC depicted in Figures 9b, 9d and 9f respectively.

Consider a set $U = \{\overrightarrow{u_1}, \ldots, \overrightarrow{u_x}\}$ of vectors where each $\overrightarrow{u_i}$ $(1{\leq}i{\leq}x)$ has its specific dimension noted here as α_i. For the particular case where property p returns a real number which can be positive or negative (e.g: properties p_1 and p_5), the following procedure is proposed for deriving the MC $\delta_p(\overrightarrow{u_i})$ associated with every vector $\overrightarrow{u_i} = [u_{i1}, \ldots, u_{i\alpha_i}]$ according to property p:

1) Generate the pseudo MC associated with $\overrightarrow{u_i}$ following Figure 4.
2) Use Equation 3 for splitting the interval [0, 180] into $(N+1)$ slices $\{s_0, s_1, \ldots, s_N\}$.
3) Replace every angle by the index of the slice to which it belongs.
4) Replace $p(u_{ik})$ by its normalized value $\overline{p}(u_{ik})$ with $(1{\leq}k{\leq}\alpha_i)$ using Equation 4 following [34][19] where such a normalization was also performed.

[18] See Equation 7 of [34].
[19] See Equation 6 of [34].

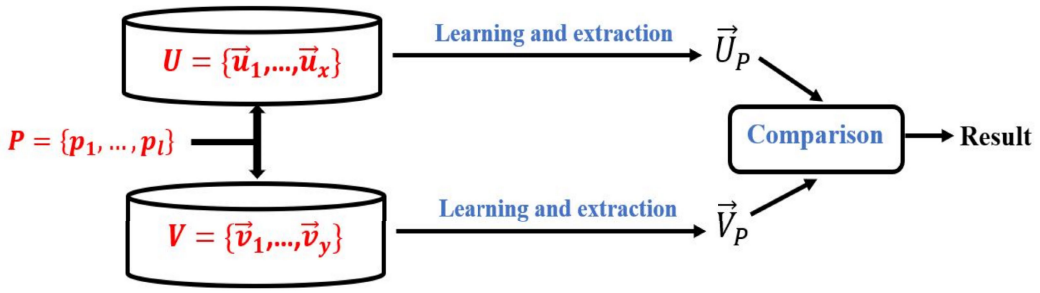

FIGURE 6. Proposed methodology for vector comparison using machine learning.

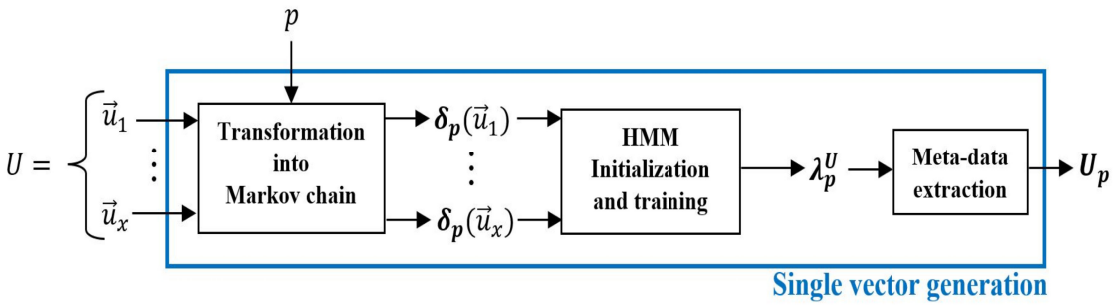

FIGURE 7. Generation of the single feature vector U_p associated with U according to p.

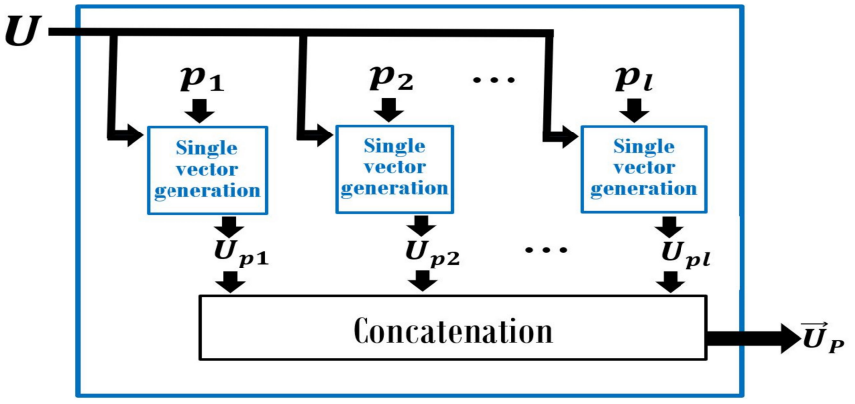

FIGURE 8. Generation of the final feature vector $\vec{U_P}$ associated with U according to P.

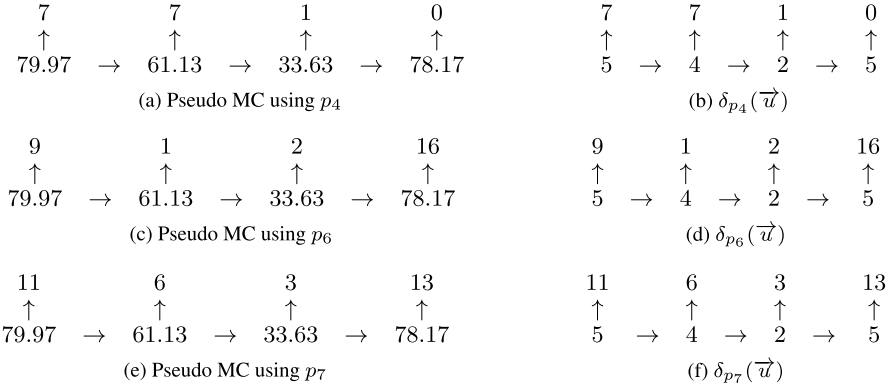

FIGURE 9. Transformation of $\vec{u} = [17, 47, 81, 20]$ into MC when p_4, p_6 and p_7 are individually selected.

Thereafter, all the $\overline{p}(u_{ik})$ belong to the interval $[-100, 100]$. If property p exclusively returns positive (resp. negative) real numbers like property p_5, all the $\overline{p}(u_{ik})$ rather belong to the interval $[0, 100]$

(resp. $[-100, 0]$) after the normalization.

$$\begin{cases} \overline{p}(u_{ik}) = \dfrac{100}{maxi} \times p(u_{ik}), \quad (1 \leq k \leq \alpha_i) \text{ where} \\ maxi = \max\limits_{1 \leq i \leq x} \left\{ \max\limits_{1 \leq k \leq \alpha_i} \{|p(u_{ik})|\} \right\} \end{cases} \tag{4}$$

5) Given a user-defined integer M, split the interval $[-100, 100]$ into $(2M + 1)$ slices $\{z_0, z_1, \ldots, z_{2M}\}$ using Equation 5. If p exclusively returns positive (resp. negative) real numbers, the interval $[0, 100]$ (resp. $[-100, 0]$) is split into $(M + 1)$ slices using only the second (resp. the third) line of Equation 5.

$$\begin{cases} z_0 = \{0\} \\ z_j = \left]\dfrac{100}{M} \times (j-1), \dfrac{100}{M} \times j\right], & (1 \leq j \leq M) \\ z_{j+M} = \left[\dfrac{-100}{M} \times (j), \dfrac{-100}{M} \times (j-1)\right[, & (1 \leq j \leq M) \end{cases} \tag{5}$$

6) Finally, replace every $\overline{p}(u_{ik})$ by the index of the slice to which it belongs.

As an example, consider now the singleton $U = \{\overrightarrow{u}\}$ where $\overrightarrow{u} = [13, -137, -72, 55, -11]$. Suppose that $N = M = 10$ for splitting the interval $[0, 180]$ into the 11 slices obtained in the former example and the interval $[-100, 100]$ into the 21 following slices: $z_0 = \{0\}$, $z_1 =]0, 10]$, $z_2 =]10, 20]$, $\ldots$, $z_{10} =]90, 100]$, $z_{11} = [-10, 0[$, $z_{12} = [-20, -10[$, $\ldots$, $z_{20} = [-100, -90[$. In these conditions:

1) The pseudo MC associated with $\overrightarrow{u}$ according to property p_1 is shown in Figure 10a.
2) When each angle is replaced by the index of the slice to which it belongs, the modified pseudo MC presented in Figure 10b is obtained.
3) The resulting values of p_1 are then replaced by their corresponding normalized values $\overline{p}_1$ calculated for $(maxi = 137)$ for deriving the modified MC presented in Figure 10x.
4) Finally, each $\overline{p}_1$ is replaced by the index of the slice to which it belongs to obtain the final MC $\delta_{p_1}(\overrightarrow{u})$ depicted in Figure 10d.

D. HMM INITIALIZATION

Consider a set $U = \{\overrightarrow{u_1}, \ldots, \overrightarrow{u_x}\}$ of vectors and a property p belonging to a user-defined set $P = \{p_1, \ldots, p_l\}$ of targeted properties. Given a small positive user-defined constant ε, the parameters of the initial HMM $\hat{\lambda}_p^U = (\hat{A}_p^U, \hat{B}_p^U, \hat{\pi}_p^U)$ associated with U according to p are calculated as described below for statistically capturing the state transitions and the symbol probability distributions from the content of the set of MC $\Delta_p^U = \{\delta_p(\overrightarrow{u_1}), \ldots, \delta_p(\overrightarrow{u_x})\}$:

1) Equation 3 enables to derive the set $\{s_0, \ldots, s_N\}$ composed of $(N + 1)$ slices. Each slice is considered here as a state of the model. Thus, the set of states is $S = \{0, 1, \ldots, N\}$.
2) The content of the set ϑ of symbols depends on the possible values returned by property p. If the returned values of p are taken from a finite set Z, then $(\vartheta = Z)$. If p returns real numbers that can be positive or negative, Equation 5 enables to derive the set $\{z_0, z_1, \ldots, z_{2M}\}$ composed of $(2M + 1)$ slices. Each slice is considered here as a symbol of the model. Thus, the set of symbols is $\vartheta = \{0, 1, \ldots, 2M\}$. If p exclusively returns positive (resp. negative) real numbers, ϑ only contains $(M + 1)$ symbols corresponding to the slices generated when only one of the two last lines of Equation 5 is used.
3) The probability of transiting from state s_i to state s_j is calculated in Equation 6. In that equation, $transit(s_i, s_j, \Delta_p^U)$ is the number of transitions from state s_i to state s_j in Δ_p^U and $transit(s_i, -, \Delta_p^U)$ is the number of transitions from state s_i to any destination in Δ_p^U.

$$\hat{A}_p^U[s_i, s_j] = \frac{transit(s_i, s_j, \Delta_p^U)}{transit(s_i, -, \Delta_p^U) + \varepsilon} \tag{6}$$

4) The probability to observe symbol z_k at state s_j is calculated in Equation 7 where $observe(z_k, s_j, \Delta_p^U)$ is the number of times symbol z_k is observed from state s_j in Δ_p^U, and $observe(-, s_j, \Delta_p^U)$ is the number of occurrences of state s_j in Δ_p^U.

$$\hat{B}_p^U[s_j, z_k] = \frac{observe(z_k, s_j, \Delta_p^U)}{observe(-, s_j, \Delta_p^U) + \varepsilon} \tag{7}$$

5) The probability of starting with state s_i is calculated in Equation 8, where $start(s_i, \Delta_p^U)$ is the number of MC in Δ_p^U starting with state s_i.

$$\hat{\pi}_p^U[s_i] = \frac{start(s_i, \Delta_p^U)}{|\Delta_p^U| + \varepsilon} \tag{8}$$

The parameters of $\hat{\lambda}_p^U$ are not probability distributions. This inconvenience is intentionally introduced by adding ε to the denominators of its various components in order to

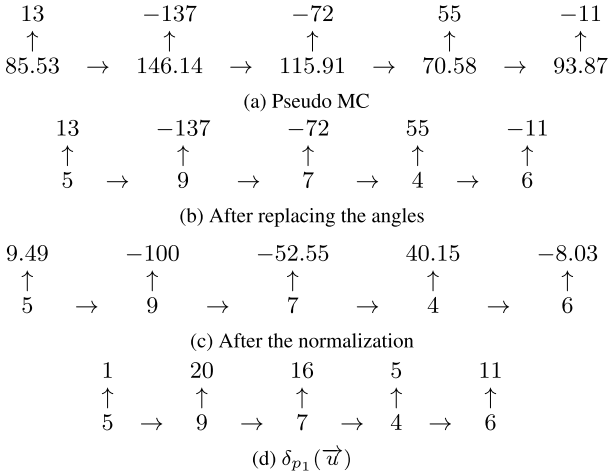

FIGURE 10. Construction of the MC associated with $\overrightarrow{u} = [13, -137, -72, 55, -11]$ according to p_1.

avoid eventual divisions by zero and zero probabilities in the initial HMM. We experimentally fixed $\varepsilon = 1.0$ and an equitable redistribution of the missing quantity is applied to each element of each line in the model $\hat{\lambda}_p^U$ to obtain the readjusted initial model $\bar{\lambda}_p^U = (\bar{A}_p^U, \bar{B}_p^U, \bar{\pi}_p^U)$ whose parameters are calculated in Equation 9.

$$\begin{cases} \bar{A}_p^U[s_i, s_j] = \hat{A}_p^U[s_i, s_j] + \dfrac{1}{N+1}\left(1 - \sum_{h=0}^{N} \hat{A}_p^U[s_i, s_h]\right) \\ \bar{B}_p^U[s_j, z_k] = \hat{B}_p^U[s_j, z_k] + \dfrac{1}{|\vartheta|}\left(1 - \sum_{h=0}^{|\vartheta|-1} \hat{B}_p^U[s_j, z_h]\right) \\ \bar{\pi}_p^U[s_i] = \hat{\pi}_p^U[s_i] + \dfrac{1}{N+1}\left(1 - \sum_{h=0}^{N} \hat{\pi}_p^U[s_h]\right) \end{cases} \tag{9}$$

E. HMM TRAINING

If $|U| = 1$, then the readjusted initial HMM $\bar{\lambda}_p^U$ is trained using the *Baum-Welch* algorithm for one single sequence, otherwise $\bar{\lambda}_p^U$ is trained by the same algorithm for multiple sequences. In both situations, the resulting final HMM is $\lambda_p^U = (A_p^U, B_p^U, \pi_p^U)$ and the training sequences are the sequences of symbols appearing in Δ_p^U. For example, the initial HMM associated with the set $\{\delta_{p_1}(\overrightarrow{u})\}$ presented in Figure 10d will be trained with the following sequence of symbols: 1, 20, 16, 5, 11.

F. META-DATA EXTRACTION

Consider a set $U = \{\overrightarrow{u_1}, \ldots, \overrightarrow{u_x}\}$ of vectors and a property p belonging to a user-defined set $P = \{p_1, \ldots, p_l\}$ of targeted properties. When the values of p are taken from the finite set $Z = \vartheta = \{z_0, z_1, \ldots, z_{|\vartheta|-1}\}$, the single feature vector $U_p = [U_p^0, U_p^1, \ldots, U_p^{|\vartheta|-1}]$ associated with U according to p is derived from λ_p^U by analyzing its behavior regarding each symbol analogically to [9][20] where the authors realized human activity recognition using HMM. More precisely, U_p^k $(0 \leq k \leq |\vartheta| - 1)$ is considered as the overall proportion of time spent by λ_p^U observing symbol z_k after a sufficiently long time, irrespective of the state from which this observation is realized. In our context, this roughly corresponds to the rate of apparition of a specific value of property p (corresponding to symbol z_k) in the training data. In order to compute the value of U_p^k, the overall proportion of time spent by λ_p^U observing symbol z_k from each state s_i after a sufficiently long time is first evaluated as follows:

1) Evaluate the overall proportion of time spent by λ_p^U in state s_i after a sufficiently long time. This proportion is given by the i^{th} component $\varphi_p^U[s_i]$ of the stationary distribution of λ_p^U.
2) Multiply the result obtained at step 1 by the probability of observing z_k from state s_i which is $B_p^U[s_i, z_k]$.

[20]See Section IV-E and Equation 7 of [9].

The value of U_p^k is finally obtained by repeating this process for every state s_i and summing the resulting proportions as shown in Equation 10.

$$\begin{cases} U_p = [U_p^0, U_p^1, \ldots, U_p^{|\vartheta|-1}] \quad \text{where} \\ U_p^k = \sum_{i=0}^{N} \left(\varphi_p^U[s_i] \times B_p^U[s_i, z_k]\right) \quad \text{with } (0 \leq k \leq |\vartheta| - 1) \end{cases} \tag{10}$$

If p returns a real number, in which case the vectors in U have been normalized using the value *maxi* calculated in Equation 4, the value of *maxi* is inserted as the last additional component of U_p in order to consider the effect of this normalization as shown in Equation 11.

$$\begin{cases} U_p = [U_p^0, U_p^1, \ldots, U_p^{|\vartheta|-1}, maxi] \quad \text{where} \\ U_p^k = \sum_{i=0}^{N} \left(\varphi_p^U[s_i] \times B_p^U[s_i, z_k]\right) \quad \text{with } (0 \leq k \leq |\vartheta| - 1) \end{cases} \tag{11}$$

Given that the proposed approach is applicable to finite sets of vectors, we conventionally adopt Property 1 for deriving the single feature vector associated with the empty set.

Property 1: The single feature vector associated with $\emptyset$ according to any property p is $\emptyset_p = \overrightarrow{0} \in \mathbb{R}^{|\vartheta|}$.

The fixed size of each single feature vector is the number of symbols of the corresponding HMM. A symbol is a discrete information describing a process and that can be '*observed*' by a HMM in order to capture the overall '*behavior*' of this process after a sufficiently long time. Consequently, a low number of symbols may induce a lack of information during the model training, which leads to an incomplete behavior description of the process. On the contrary, a high number of symbols may lead to overfitting during the model training and generate an inconsistent behavior description of the process.

G. FINAL FEATURE VECTOR GENERATION

The final feature vector $\overrightarrow{U_P}$ associated with U according to all the properties in the user-defined set $P = \{p_1, \ldots, p_l\}$ of targeted properties is constructed by sequentially concatenating in this order the components of the single feature vectors $U_{p_1}, \ldots, U_{p_l}$ as described in Algorithm 1 where β_i is the dimension of U_{p_i} with $(1 \leq i \leq l)$.

H. VECTOR SETS COMPARISON

Consider two sets $U = \{\overrightarrow{u_1}, \ldots, \overrightarrow{u_x}\}$ and $V = \{\overrightarrow{v_1}, \ldots, \overrightarrow{v_y}\}$ of vectors, each vector $\overrightarrow{u_i}$ $(1 \leq i \leq x)$ and $\overrightarrow{v_j}$ $(1 \leq j \leq y)$ having its specific dimension. Given a user-defined set $P = \{p_1, \ldots, p_l\}$ of targeted properties and any existing distance (resp. similarity) measure d between two vectors having the same dimension, we define in Equation 12 the corresponding *σ-distance* (resp. *σ-similarity*) noted σ_P^d between U and V obtained by computing the distance/similarity d their respective associated feature vectors $\overrightarrow{U_P}$ and $\overrightarrow{V_P}$.

$$\sigma_P^d(U, V) = d(\overrightarrow{U_P}, \overrightarrow{V_P}) \tag{12}$$

TABLE 4. *σ-Euclidean* and *σ-Manhattan* distances between the singletons $U = \{\overrightarrow{u_1}\}$ and $V = \{\overrightarrow{v_1}\}$ according to P, with $\overrightarrow{u_1} = [17, 47, 81, 20]$ and $\overrightarrow{v_1} = [13, -137, -72, 55, -11]$.

P	$\overrightarrow{U_P}$	$\overrightarrow{V_P}$	$\sigma_P^{d_1}$	$\sigma_P^{d_2}$
$\{p_1\}$	$[0,0,0,1,0,0,0,0,0,0,0,0,0,0,$ $0,0,0,0,0,0,0,0,81]$	$[0,0.04,0,0,0,0.22,0,0,0,0,0,0.37,$ $0,0,0,0,0.2,0,0,0,0.17,137]$	56.01	58.0
$\{p_4\}$	$[0.65,0.17,0,0,0,0,0,0.18,0,0]$	$[0,0.7,0.09,0.03,0,0.12,0,0.06,0,0]$	0.85	1.52
$\{p_6\}$	$[0.12,0.17,0,0,0,0,0,0,0.06,0,0,$ $0,0,0,0,0.65]$	$[0,0.06,0.13,0,0,0,0,0.1,0.03,0,0,$ $0,0.68,0,0,0]$	0.96	1.81
$\{p_7\}$	$[0,0,0.17,0,0,0.12,0,0,0,0,$ $0.06,0,0.65,0,0,0]$	$[0,0.13,0.1,0,0,0.06,0,0,0,0,0.03,$ $0,0,0,0,0.68]$	0.95	1.61

TABLE 5. *σ-Euclidean* and *σ-Manhattan* distances between the sets $U' = \{\overrightarrow{u_1}, \overrightarrow{u_2}\}$ and $V' = \{\overrightarrow{v_1}, \overrightarrow{v_2}\}$ according to P, with $\overrightarrow{u_1} = [17, 47, 81, 20]$, $\overrightarrow{u_2} = [44, 156, 84, 548]$, $\overrightarrow{v_1} = [13, -137, -72, 55, -11]$ and $\overrightarrow{v_2} = [188, 4, 326, 142]$.

P	$\overrightarrow{U'_P}$	$\overrightarrow{V'_P}$	$\sigma_P^{d_1}$	$\sigma_P^{d_2}$
$\{p_1\}$	$[0,0.3,0.4,0.1,0,0,0,0,0,0,$ $0.2,0,0,0,0,0,0,0,0,0,0,548]$	$[0,0.13,0.06,0,0,0.35,0.06,0,0,0,0.2,$ $0.08,0,0.08,0,0.04,0,0,0,0,0,326]$	222	223.23
$\{p_4\}$	$[0.63,0.09,0,0,0.09,0,0.03,$ $0.09,0.09,0]$	$[0,0.5,0,0,0.5,0,0,0,0,0]$	0.87	1.66
$\{p_6\}$	$[0.06,0.08,0,0,0.11,0,0,0,$ $0.03,0,0.03,0,0,0,0.08,0.61]$	$[0,0.06,0.06,0,0.27,0,0,0.09,$ $0.03,0,0.06,0,0.08,0,0.34,0]$	0.7	1.37
$\{p_7\}$	$[0,0,0,0,0,0,0,0,0,0,0,0,$ $1,0,0,0]$	$[0.27,0.06,0.09,0,0,0.06,0,0,0.06,$ $0,0.03,0,0.34,0,0,0.08]$	0.73	1.31

Algorithm 1 *FinalFeatureVector*

Inputs: $U, P = \{p_1, \ldots, p_l\}, \{\lambda_{p_1}^U, \ldots, \lambda_{p_l}^U\}$
Output: $\overrightarrow{U_P}$

```
k ← 1
for (i ← 1; (i ≤ l); i ← i + 1) do
    Compute(U_{p_i})
    for (j ← 1; (j ≤ β_i); j ← j + 1) do
        U_P[k] ← U_{p_i}[j]
        k ← k + 1
    end for
end for
return U_P
```

I. PRACTICAL EXAMPLES

To illustrate how the proposed approach can be used in practice, we first computed the *σ-Euclidean* and the *σ-Manhattan* distances between the singletons $U = \{\overrightarrow{u_1}\}$ and $V = \{\overrightarrow{v_1}\}$, where $\overrightarrow{u_1} = [17, 47, 81, 20]$ and $\overrightarrow{v_1} = [13, -137, -72, 55, -11]$ are the vectors used for the examples presented in Section IV-C. The comparison is respectively realized according to the sets $\{p_1\}$, $\{p_4\}$, $\{p_6\}$ and $\{p_7\}$ of properties. For this example, we also fixed $N = M = 10$ as it was the case for the examples of Section IV-C. The final feature vectors and the comparison results are presented in Table 4.

In identical conditions, we also computed the *σ-Euclidean* and the *σ-Manhattan* distances between the sets $U' = \{\overrightarrow{u_1}, \overrightarrow{u_2}\}$ and $V' = \{\overrightarrow{v_1}, \overrightarrow{v_2}\}$, where $\overrightarrow{u_1}$ and $\overrightarrow{v_1}$ are unchanged, while $\overrightarrow{u_2} = [44, 156, 84, 548]$ and $\overrightarrow{v_2} = [188, 4, 326, 142]$ are the two vectors taken as examples in the paper Introduction. The final feature vectors and the comparison results are presented in Table 5. All the comparison results presented in Tables 4 and 5 are consistent since they represent the *Euclidean* and the *Manhattan* distances between U and V calculated according to different sets of criteria. The generation of each final feature vector computed in this section experimentally required less than 200 ms.

V. EXPERIMENTAL RESULTS

As it can be observed in Equation 12, the current work does not propose any new similarity or distance measure for comparing vectors. It rather enhances existing measures by giving them the ability:

1) To perform the comparison according to a specific set P of targeted properties.
2) To compare two finite sets of vectors, each containing vectors of various dimensions.

The goal of this section is to demonstrate that there are situations where it is preferable to compare vectors according to other criteria than the exact values of their components and to evaluate the performances of the proposed approach in these conditions. In order to show how the proposed approach can efficiently enhance existing techniques when the suitable set of targeted properties is selected, flat classification experiments were conducted on three synthetic datasets especially constructed for this purpose.

TABLE 6. Properties in *Modulo10*. For each vector $\vec{u} = [u_1, \ldots, u_{10}]$, the item located at line c and column u_k defines the value of ($|u_k| \bmod 10$) in class c.

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
1	0	1	2	3	4	5	6	7	8	9
2	2	4	9	8	5	1	7	0	3	6
3	5	3	6	0	1	8	2	4	9	7
4	1	8	0	7	9	4	5	2	6	3
5	4	3	7	5	1	0	2	9	6	8
6	7	0	4	9	3	8	5	1	2	6
7	3	9	6	8	2	1	7	0	5	4
8	6	2	8	1	5	9	0	7	4	3
9	1	6	3	0	8	7	9	4	2	5
10	9	1	3	5	7	6	0	8	4	2

TABLE 7. Properties in *OddOrEven*. For each vector $\vec{u} = [u_1, \ldots, u_{10}]$, the item located at line c and column u_k indicates if $|u_k|$ is even e or odd o in class c.

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
1	e	e	o	o	e	e	o	o	e	e
2	o	e	o	e	o	o	o	e	e	o
3	e	o	o	e	o	o	e	o	e	o
4	o	o	o	o	e	e	o	o	o	e
5	e	o	e	e	e	e	o	e	o	o
6	e	o	e	o	o	o	o	e	e	e
7	o	o	o	e	o	o	e	e	e	e
8	o	o	o	o	o	e	e	e	e	e
9	e	e	e	e	o	o	e	o	o	e
10	o	e	e	e	o	o	e	e	e	o

TABLE 8. Properties in *FirstOrNot*. For each vector $\vec{u} = [u_1, \ldots, u_{10}]$, the item located at line c and column u_k indicates if $|u_k|$ is a first number $+$ or not $-$ in class c.

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
1	$+$	$+$	$-$	$-$	$+$	$+$	$-$	$-$	$+$	$+$
2	$-$	$+$	$-$	$+$	$-$	$-$	$-$	$+$	$+$	$-$
3	$+$	$-$	$-$	$+$	$-$	$-$	$+$	$-$	$+$	$-$
4	$-$	$-$	$-$	$-$	$+$	$+$	$-$	$-$	$-$	$+$
5	$+$	$-$	$+$	$+$	$+$	$+$	$-$	$+$	$-$	$-$
6	$+$	$-$	$+$	$-$	$-$	$-$	$-$	$+$	$+$	$+$
7	$-$	$-$	$-$	$+$	$-$	$-$	$+$	$+$	$+$	$+$
8	$-$	$-$	$-$	$-$	$-$	$+$	$+$	$+$	$+$	$+$
9	$+$	$+$	$+$	$+$	$-$	$-$	$+$	$-$	$-$	$+$
10	$-$	$+$	$+$	$+$	$-$	$-$	$+$	$+$	$+$	$-$

A. EXPERIMENTAL DATABASES

The three synthetic datasets are *Modulo10*, *OddOrEven* and *FirstOrNot*. Each dataset contains 10 classes composed of 100 vectors, each having 10 components. The components of the vectors in *Modulo10* are integers from the interval $[-200, 200]$, while those of the two remaining datasets are integers from the interval $[0, 200]$. Tables 6 to 8 describe the properties verified by the components of each vector for each class respectively in *Modulo10*, *OddOrEven* and *FirstOrNot*. All the data used during the current experiments are available online as '*.arff*' files[21] taken as input by the soft WEKA [35].

B. EXPERIMENTAL SETTINGS

The classification experiments performed in this work were realized on a personal computer having 16 GB of main memory and the following processor: *Intel(R) Core(TM) i7-8665U CPU @ 1.90 GHz 2.11 GHz*.

The maximum number of iterations $\gamma = 100$ was selected here for the *Baum-Welch* algorithm[22] and the constant $r = 100$ was selected here for computing the stationary distributions of our HMMs[23] following [9].

Each vector $\vec{u}$ of each dataset was considered as the singleton $U = \{\vec{u}\}$. Given that the three synthetic experimental datasets contain vectors having components belonging to $\mathbb{N}$, Equation 10 was used for the generation of the final feature vectors.

It is not systematically necessary to determine the most relevant and informative content of the set P of targeted properties when using the proposed approach since the comparison result always remains consistent according to P, whatever it contains. It is only the final goal of the comparison which indicates the necessity of determining the best content of P. As an example, for pure decision support as comparing sets of items (e.g: *cars*) represented by vector data in order to select the set which matches the best with some user-defined preferences (e.g: *number of doors* ≤ 4, *engine power* ≥ 50, etc.), it is not necessary to determine the best content of P. Indeed, the user-defined preferences fully determine the content of P in that context. However, the discovery of the best content of P becomes necessary and challenging for classification purposes for example. It is for this reason that the sets $\{p_4\}$, $\{p_6\}$, $\{p_7\}$, $\{p_4, p_6\}$, $\{p_4, p_7\}$, $\{p_6, p_7\}$ and $\{p_4, p_6, p_7\}$ have been objectively selected for the current experiments. Testing several sets of properties for each experimental dataset enables to observe how the selected set of properties impacts the classification results. The value ($N = 20$) was selected for splitting the interval $[0, 180]$ using Equation 3. All our codes were written in C language and are also available online [21].

C. CLASSIFICATION RESULTS

Classification experiments were realized in WEKA [35] through a 10 fold cross-validation. The four following classifiers were selected with their default parameters in WEKA for this purpose, their corresponding names in WEKA are shown in brackets:

1) *Nearest Neighbor* (IBk) with $k = 1$, which was used for the *Euclidean* and for the *Manhattan* distances.
2) *Support Vector Machines* (SMO) with the following parameters: $c = 1, \varepsilon = 10^{-12}$ and *kernel* = polynomial kernel.
3) *Decision trees* (J48) with the following parameters: *confidence factor* = 0.25, *Minimum number of objects* = 2, *Binary splits* = *False*.

The selected classifiers have been previously used in [9] and [34] for analogue purposes. In particular, the use of the *Nearest Neighbor* classifier is important here because it enables to evaluate the behavior of existing distances when

[21] https://webperso.etis-lab.fr/sylvain.iloga/Vectors/index.html

[22] See Section III-C of [9].

[23] See Section V-B of [9].

TABLE 9. Best classification results considering the four selected classifiers when the final feature vectors are used. Accuracies are in (%). The best accuracies are in bold in each column.

P	Modulo10	OddOrEven	FirstOrNot
$\{p_4\}$	74.4	24.9	18.2
$\{p_6\}$	82.4	**92.9**	49.1
$\{p_7\}$	38.7	34.9	**93.1**
$\{p_4, p_6\}$	91.9	90.5	39.6
$\{p_4, p_7\}$	72.4	32.2	90.4
$\{p_6, p_7\}$	82.9	91.2	90.3
$\{p_4, p_6, p_7\}$	**93.0**	91.1	90.0

TABLE 10. Comparison of the best classification performances when the original vectors are used and when the final feature vectors are used. Accuracies are in (%). The best accuracies are in bold.

Dataset	Final feature vectors	Original vectors	Accuracy gain
Modulo10	**93.0**	24.1	+68.9
OddOrEven	**92.9**	10.7	+82.3
FirstOrNot	**93.1**	15.8	+77.3

they compare original vectors on the one hand, and when they compare the final feature vectors generated by the proposed approach on the other hand. Classification results are presented in Table 9 where each cell contains the best classification performance for the considered dataset and considering all the four selected classifiers. These results experimentaly demonstrate that the final feature vectors generated by the proposed approach according the sets of selected properties enable to obtain good performances for the three experimental datasets.

D. COMPARISONS

In order to compare the proposed approach to existing techniques, we evaluated the flat classification performances when each original vector $\overrightarrow{u}$ of each dataset is directly used as input for the classifier. The resulting best classification performances are extremely poor for the three synthetic datasets. When the best accuracies obtained by existing techniques are compared to those exhibited by the proposed approach in Table 10, we observe that the proposed approach outperforms existing techniques with accuracy gains reaching +82.3%.

E. TIME COST

1) THEORETICAL TIME COST

According to the methodology presented in Figure 8, the time cost of the proposed approach is approximated by the time required for the generation of one single feature vector, multiplied by the number l of properties in $P = \{p_1, \ldots, p_l\}$. For each property $p \in P$, Figure 7 reveals that the generation of each single feature vector is composed of the three following steps:

1) *Transformation into MC*: Let us note ξ the time required for an angle computation and κ the maximum time required for the evaluation of any property $p \in P$. The transformation into MC roughly only requires n computations of angles (states) and n evaluations of property p (symbols) for a vector $\overrightarrow{u} = [u_1, \ldots, u_n]$. Thus for one vector, this step requires $n.(\xi + \kappa)$. In these conditions, the transformation of the vectors belonging to a set $U = \{\overrightarrow{u_1}, \ldots, \overrightarrow{u_x}\}$ into MC runs in $\theta\left[(\xi + \kappa).\left(\sum_{i=1}^{x} \alpha_i\right)\right]$ where α_i is the dimension of $\overrightarrow{u_i}$.
2) *HMM initialization and training*: The time required for initializing a model was experimentally very low. Thus this step is dominated by the model training phase, which according to Section III-C, runs in $\theta\left[\gamma.(N+1)^2.\left(\sum_{i=1}^{x} \alpha_i\right)\right]$ where N is the user-defined constant used for splitting the interval $[0, 180]$ into $(N + 1)$ slices and γ is the user-defined maximum number of iterations of the *Baum-Welch* algorithm.
3) *Meta-data extraction*: The main time consuming operation realized during this step is the computation of the stationary distribution of the HMM which runs in $\theta\left[r.(N+1)^3\right]$ as stated in Section III-D.

When these 3 partial results are gathered, we deduce that the time required for the single vector generation is $\theta\left[r.(N+1)^3 + \left(\xi + \kappa + \gamma.(N+1)^2\right).\left(\sum_{i=1}^{x} \alpha_i\right)\right]$. When N is high, this time cost can simply be reduced to $\theta\left[r.N^3 + \left(\xi + \kappa + \gamma.N^2\right).\left(\sum_{i=1}^{x} \alpha_i\right)\right]$. The time needed for the generation of the final feature vector $\overrightarrow{U_P}$ is obtained by multiplying this expression by l as shown in Equation 13.

$$Time(\overrightarrow{U_P}) = \theta\left[l.r.N^3 + l.\left(\xi + \kappa + \gamma.N^2\right).\left(\sum_{i=1}^{x} \alpha_i\right)\right] \tag{13}$$

This time complexity can become high if the values of the parameters $\xi, \kappa, \alpha_i, r, \gamma$ and N are also high. The parameters ξ, κ and α_i are not user-dependent. Therefore, they cannot be modified for reducing the time cost. However, the values of r, γ and N which are user-defined can be gradually reduced without necessarily having an important negative impact on the final comparison results. Indeed:

1) If the stationary distribution is discovered after r_0 iterations with $(r_0 < r)$, this distribution will remain unchanged during the $(r - r_0)$ remaining iterations.
2) Similarly, the *Baum-Welch* algorithm may discover a model very close to the local optimum after γ_0 iterations with $(\gamma_0 < \gamma)$, in which case it is not absolutely necessary to perform the remaining $(\gamma - \gamma_0)$ iterations.

2) EXPERIMENTAL TIME COST

As it can be observed in Table 11 the experimental time cost required by the proposed approach is reasonable. This table shows the maximum time costs (in milliseconds) required for each main step of the proposed approach in each experimental dataset. Table 11 reveals that the most time consuming steps of the proposed approach are:

1) The '*HMM initialization and training*' whose time cost can be reduced if the *Baum-Welch* algorithm is executed in parallel.

TABLE 11. Maximum experimental time costs (in milliseconds) required for each main step of the proposed approach in each experimental dataset, with **Step 1** = *'Transformation into Markov chain'*, **Step 2** = *'HMM initialization and training'* and **Step 3** = *'Feature vector generation'*.

Dataset	P	Step 1	Step 2	Step 3
Modulo10	$\{p_4\}$	1	65	71
OddOrEven	$\{p_6\}$	1	80	73
FirstOrNot	$\{p_7\}$	1	66	65

2) The '*Feature vector generation*' whose duration can be reduced if the stationary distribution is discovered after a low number of iterations as explained at the end of Section V-E1.

F. MAIN ASSETS

The technique proposed in the current paper:

1) Enables to compare two finite sets of vectors.
2) Enables to compare vectors having different dimensions without distorting the reality, unlike *ZP* and *DR* techniques.
3) Requires the explicit specification of a finite set P of targeted properties according to which the comparison is performed. No existing technique offers this option.
4) Unlike *SNN*, it takes a reasonable time for associating a HMM λ_p^U with a set U of vectors according to each property $p \in P$. The parameters of this HMM can be easily modified and the resulting single feature vector $\overrightarrow{U_p}$ is interpretable. Indeed, the k^{th} component U_p^k of $\overrightarrow{U_p}$ is the overall proportion of time spent by the model λ_p^U observing the k^{th} symbol after a sufficiently long time, irrespective of the state from which this observation is realized.
5) When the suitable set of property is selected, it seriously outperforms existing techniques for the flat classification of the three experimental datasets with accuracy gains reaching $+82.3\%$.
6) Can be easily implemented in parallel in order to reduce its time cost.

VI. CONCLUSION

This paper addresses the problem of vectors comparison which is very important for popular tasks like classification and clustering. Existing distance and similarity measures used for this purpose are unsuitable when the input vectors have different dimensions. *ZP* and *DR* techniques are most often used in that case to augment/reduce the dimensions of the inputs vectors such that they finally have the same dimension before performing the comparison. But these techniques distort the reality. *SNN* enable to avoid the distortion of the reality, but they are limited by several factors (lots of computing resources, less interpretable outputs, etc). Additionally, the problem of comparing two finite sets of vectors remains an open field of research.

The current paper has attempted to overcome these drawbacks by proposing a customizable technique based on HMM for comparing two finite sets of vectors, each set containing vectors having different dimensions, while precising the set of targeted properties on which the comparison should be performed. Flat classification experiments conducted on three online available custom datasets demonstrated that when the suitable set of targeted properties is selected, the proposed approach outperforms existing techniques with accuracy gains reaching $+82.3\%$. The following perspectives can be considered in future work:

1) Future work can analyze the impact of the gradual modification of the user-defined parameters r, γ, N and M on the performances of the proposed approach.
2) Future work must focus on the objective discovery of the most relevant and informative set of properties by analyzing the vector components using machine learning or data mining techniques when it is required by the final goal of the comparison. Given two finite vector sets U and V, future work can analyze the content of U in order to discover of the subset P_U of targeted properties that is suitable for the characterization of the vectors in U. The same process will be used for the discovery of the subset P_V of targeted properties that is suitable for the characterization of the vectors in V. Finally, the comparison between U and V will be done using the proposed approach according to the set $P = \left(P_U \bigcup P_V\right)$.
3) The time cost of the proposed approach can be reduced in future work if the single feature vectors are generated in parallel. More precisely, if the targeted set of properties is $\{p_1, p_2, \ldots, p_L\}$, then L processors $\{proc_1, proc_2, \ldots, proc_L\}$ can be used for this purpose, each processor $proc_k (1 \leq k \leq L)$ being responsible of the generation of the single feature vector associated with property p_k. This time cost can be further reduced if a parallel version of the *Baum-Welch* algorithm is additionally executed. This can be implemented using *Message Passing Interface* (MPI) following [36] or using a *Field-Programmable Gate Array* (FPGA) chip following [37].
4) A taxonomy is an efficient navigating and browsing mechanism of a dataset. It organizes the content of the dataset into a hierarchy where broad concepts are at the top and more specific concepts are further down. Future work must explore the possibility of designing datasets taxonomies based on the feature vectors generated by the proposed approach. More formally, given a dataset composed of many classes, a unique feature vector can be generated for each class using the proposed approach. The resulting feature vectors can be finally taken as input by a hierarchical clustering algorithm in order to generate a taxonomy of the dataset.
5) Future work must focus on the implementation of an extension of the proposed approach that will consider vectors having nominal components or a mixture of numeric and nominal components.

REFERENCES

[1] K. Jajuga, A. Sokolowski, and H.-H. Bock, *Classification, Clustering, and Data Analysis: Recent Advances and Applications*. Berlin, Germany: Springer-Verlag, 2012.

[2] G. W. Milligan and S. C. Hirtle, "Clustering and classification methods," in *Handbook of Psychology: Research Methods in Psychology*, vol. 2. New York, NY, USA: Wiley, 2003, pp. 165–186.

[3] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *City*, vol. 1, no. 2, p. 1, 2007.

[4] M. A. Zaid, "Correlation and regression analysis textbook," *The Statistical, Economic and Social Research and Training Centre for Islamic Countries (SESRIC), Diplomatic Site*, vol. 6450. Jeddah, Kingdom of Saudi Arabia: Organisation Islamic Cooperation (OIC), 2015.

[5] Y. A. Al-Jawhar, K. N. Ramli, M. A. Taher, N. S. M. Shah, L. Audah, and M. S. Ahmed, "Zero-padding techniques in OFDM systems," *Int. J. Electr. Eng. Informat.*, vol. 10, no. 4, pp. 704–725, 2018.

[6] S. Ayesha, M. K. Hanif, and R. Talib, "Overview and comparative study of dimensionality reduction techniques for high dimensional data," *Inf. Fusion*, vol. 59, pp. 44–58, Jul. 2020.

[7] F. Anowar, S. Sadaoui, and B. Selim, "Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100378.

[8] D. Chicco, "Siamese neural networks: An overview," in *Artificial Neural Networks. Methods in Molecular Biology*, vol.2190, H. Cartwright, Ed. New York, NY, USA: Humana 2021, pp. 73–94, doi: 10.1007/978-1-0716-0826-5_3.

[9] S. Iloga, A. Bordat, J. Le Kernec, and O. Romain, "Human activity recognition based on acceleration data from smartphones using HMMs," *IEEE Access*, vol. 9, pp. 139336–139351, 2021.

[10] H. Michael. (2020). *Cluster Analysis: Basic Concepts and Algorithms*. [Online]. Available: https://mhahsler.github.io/Introduction_to_Data_Mining_R_Examples/slides/chap7_basic_cluster_analysis.pdf

[11] S. Chen, B. Ma, and K. Zhang, "On the similarity metric and the distance metric," *Theor. Comput. Sci.*, vol. 410, nos. 24–25, pp. 2365–2376, May 2009.

[12] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, Nov. 2000.

[13] G. Kraemer, M. Reichstein, and M. D. Mahecha, "dimRed and coRanking—Unifying dimensionality reduction in R," *R J.*, vol. 10, no. 1, pp. 342–358, 2018.

[14] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *J. Mach. Learn. Res.*, vol. 16, pp. 2859–2900, 2015.

[15] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani, and A. Hooman, "An overview of principal component analysis," *J. Signal Inf. Process.*, vol. 4, no. 3B, p. 173, 2013.

[16] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular value decomposition and principal component analysis," in *A Practical Approach to Microarray Data Analysis*. Cham, Switzerland: Springer, 2003, pp. 91–109.

[17] T. K. Landauer, D. S. McNamara, S. Dennis, and W. Kintsch, *Handbook of Latent Semantic Analysis*. London, U.K.: Psychology Press, 2013.

[18] X. He and P. Niyogi, "Locality preserving projections," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 16, no. 16, 2004, pp. 153–160.

[19] A. Hyvärinen, J. Hurri, and P. O. Hoyer, "Independent component analysis," in *Natural Image Statistics*. Cham, Switzerland: Springer, 2009, pp. 151–175.

[20] A. J. Izenman, "Linear discriminant analysis," in *Modern Multivariate Statistical Techniques*. Cham, Switzerland: Springer, 2013, pp. 237–280.

[21] P. J. Huber, "Projection pursuit," *Ann. Statist.*, vol. 13, no. 2, pp. 435–475, 1985.

[22] D. DeMers and G. W. Cottrell, "Non-linear dimensionality reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 580–587.

[23] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 199, pp. 583–5887.

[24] M. A. A. Cox and T. F. Cox, "Multidimensional scaling," in *Handbook of Data Visualization*. Cham, Switzerland: Springer, 2008, pp. 315–347.

[25] A. Saxena, A. Gupta, and A. Mukerjee, "Non-linear dimensionality reduction by locally linear isomaps," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2004, pp. 1038–1043.

[26] X. Geng, D.-C. Zhan, and Z.-H. Zhou, "Supervised nonlinear dimensionality reduction for visualization and classification," *IEEE Trans. Syst., Man, Cybern., B Cybern.*, vol. 35, no. 6, pp. 1098–1107, Dec. 2005.

[27] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[28] T. Kohonen, "Essentials of the self-organizing map," *Neural Netw.*, vol. 37, pp. 52–65, Jan. 2013.

[29] T. Kohonen, "Learning vector quantization," in *Self-Organizing Maps*. Cham, Switzerland: Springer, 2001, pp. 245–261.

[30] E. Schubert and M. Gertz, "Intrinsic t-stochastic neighbor embedding for visualization and outlier detection," in *Proc. Int. Conf. Similarity Search Appl.* Cham, Switzerland: Springer, 2017, pp. 188–203.

[31] M.-S. Yang and K. P. Sinaga, "A feature-reduction multi-view k-means clustering algorithm," *IEEE Access*, vol. 7, pp. 114472–114486, 2019.

[32] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[33] J. Kang. (2021). *Stat 243: Stochastic Process*. [Online]. Available: https://bookdown.org/jkang37/stochastic-process-lecture-notes/lecture09.html

[34] S. Iloga, O. Romain, and M. Tchuenté, "An accurate HMM-based similarity measure between finite sets of histograms," *Pattern Anal. Appl.*, vol. 22, no. 3, pp. 1079–1104, Aug. 2019.

[35] H. W. Ian and F. Eibe. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. [Online]. Available: http://weka.sourceforge.net/

[36] J. B. B. Noussi, M. T. Tchendji, and S. Iloga, "Parallel HMM-based similarity between finite sets of histograms," in *Proc. 4th Conf. Recherches Informatique (CRI)*, Yaoundé, Cameroon, 2019, pp. 1–8.

[37] A. López-López A. Espinosa-Manzo and M. O. Arias-Estrada, "Implementing hidden Markov models in a hardware architecture," in *Proc. Int. Meeting Comput. Sci. (ENC)*, Aguascalientes, Mexico, vol. 2, 2001, pp. 1007–1016.

EMMANUEL DELI MADIGA received the degree in renewable energy engineering from École Nationale Supérieure Polytechnique, The University of Maroua, Cameroon, in 2015, with a focus on solar photovoltaic, and the master's degree in computer science from the Faculty of Sciences, The University of Maroua, in 2022, under the supervision of Sylvain Iloga. He has been a Teacher of computer science graduated by the High Teacher's Training College, The University of Maroua, since 2016. His current research interest includes data modeling using hidden Markov models.

SYLVAIN ILOGA received the Ph.D. degree in computer science from the University of Yaoundé 1, Cameroon, in January 2018. Since January 2010, he has been a Teacher with the Department of Computer Science, High Teachers' Training College, The University of Maroua, Cameroon. From September 2017 to August 2019, he completed a research and teaching internship with the Department of Electronic Engineering and Industrial Computing, IUT of Cergy-Pontoise, Neuville University, France. In May 2018, he was promoted to the rank of Lecturer in Cameroon. Subsequently, he obtained his qualification for the functions of Lecturer in France, in January 2019, section 27 (computer science). Since October 2023, he has been a Teacher with ESIEE-IT, CCI Paris île-de-France, France. His current research interests include the design of taxonomies for hierarchical classification, sequential data mining, machine learning using hidden Markov models, and the implementation of reconfigurable architectures based on the FPGA technology.