# Histogram-based comparison of metric spaces using HMMs

Sylvain Iloga

Evolutionary Intelligence
https://doi.org/10.1007/s12065-022-00773-4

RESEARCH PAPER

# Histogram-based comparison of metric spaces using HMMs

**Sylvain Iloga**[1,2,3]

Received: 29 December 2021 / Revised: 12 August 2022 / Accepted: 17 August 2022


**Abstract**
Comparison is a powerful tool for decision support. Many measures have been proposed for comparing two metric spaces, but they do not consider the data dispersion in each metric space. Furthermore, no dedicated measure has yet been proposed for comparing two metric spaces containing elements belonging to various subtypes of a global datatype. In order to attenuate the aforementioned limitations, this paper proposes a new technique for comparing two metric spaces. More formally, given a metric space $X$, histograms are used for performing a gradual analysis of the data dispersion inside the neighborhood of each element of $X$. This is a refinement of the neighborhoods' analysis realized in an existing work. Then, another existing technique is used for associating one hidden Markov model $\lambda_X$ with $X$ such that $\lambda_X$ learns the bin values and the visual shapes of the histograms derived from the instances in $X$. Meta-data derived from $\lambda_X$ are then saved as the components of a descriptor vector $\vec{X}$ associated with $X$. Finally, the comparison between two metric spaces is performed through the comparison of their respective associated descriptor vectors using existing distance or similarity measures between two vectors. The proposed approach inherits the accuracy and the efficiency of the existing techniques on which it relies. Therefore, the experiments realized in this paper are only intended to show how it can be used for comparing particular metric spaces containing geolocations or stars in the celestial sphere.

**Keywords** Metric spaces · Comparison of datasets · Distance between datasets · Histograms · Hidden Markov Models

## 1 Introduction

Human being daily faces situations for which he must make choices. To achieve this, he generally makes comparisons because comparison is a powerful tool for decision support. The comparison usually consists either in calculating the similarity between two elements, or calculating the distance separating the two elements. For this reason, several similarity and distance measures between two elements belonging to the same datatype have been proposed. Given a set $\beta$, a *distance* (also called a *metric*) is a function *dist* which assigns a positive real number $dist(x, y)$ to every pair $x, y \in \beta$ satisfying the 4 axioms [1][1].defined in Eq. 1.

$$\begin{aligned} &1.\ d(x,y) \geq 0 \\ &2.\ d(x,y) = 0 \Leftrightarrow (x = y) \\ &3.\ d(x,y) = d(y,x) \\ &4.\ d(x,z) + d(z,y) \geq d(x,y),\ \ \forall z \in \beta \end{aligned} \tag{1}$$

A similarity is a function $d$ which also outputs a real number $d(x, y)$ satisfying the 5 axioms [1][2]defined in Eq. 2.

$$\begin{aligned} &1.\ d(x,y) \geq d(y,x) \\ &2.\ d(x,x) \geq 0 \\ &3.\ d(x,x) \geq d(x,y) \\ &4.\ d(x,y) + d(y,z) \leq d(x,z) + d(y,y),\ \forall z \in \beta \\ &5.\ d(x,x) = d(y,y) = d(x,y) \Leftrightarrow (x = y) \end{aligned} \tag{2}$$

Due to the huge sizes of the data manipulated in many domains, humans must also now be able to compare not just two single elements, but two sets composed of many

✉ Sylvain Iloga
sylvain.iloga@gmail.com

[1] University of Maroua, Higher Teachers' Training College, Department of Computer Science, P.O.box 55 Maroua, Maroua, Cameroon

[2] CY Cergy Paris University, ENSEA, CNRS, ETIS UMR 8051, F-95000 Cergy, France

[3] University of Sorbonne IRD, UMMISCO, F-93143 Bondy, France

[1] See Definition 1 of [1].
[2] See Definition 2 of [1].

Published online: 02 September 2022

**Table 1** Main characteristics of the existing techniques for comparing two datasets

| Principle | Authors [Ref.] | Year | Method/Model | Type |
|---|---|---|---|---|
| Cluster-based | Shalizi [5] | 2009 | Ward's method | Generic |
| | Hahsler [4] | 2020 | Minimum, maximum, average and centroid linkages | |
| Model-based | Tatti [6] | 2007 | Frequency $f \in \mathbb{R}^n$ | Samples |
| | Facundo [7] | 2017 | Isometric copies | Generic |
| | Iloga & al. [8] | 2018 | Hidden Markov models | Histograms |
| | Iloga & al. [9] | 2020 | | Generic |
| | Iloga [10] | 2021 | | Trees |

elements. This paper exclusively deals with particular sets called metric spaces. A metric space is a pair $(\beta, d)$ where $\beta$ is a set and $d$ is a metric on $\beta$ [2][3]. The metric space $(\beta, d)$ will be noted as $\beta$ in the rest of this paper for simplicity. Given a metric space $\beta$, we will indifferently use the terms *individual*, *instance* or *element* to designate any $x \in \beta$ in the rest of this paper.

Metric spaces can be organized in two categories: a metric space $\beta$ can either be composed of elements from the same datatype $T$, or it can rather be composed of elements from various subtypes of $T$. In this second case, $\beta$ contains elements belonging to the datatype $T$, which can be viewed itself as $n \geq 2$ distinct subtypes $T_1, \ldots, T_n$. If we note $\beta_i = \{x \in \beta | sub\ type(x) = T_i\}$, then $\beta = \{\beta_1, \ldots, \beta_n\}$.

Depending on the targeted application, a given metric space can belong to both categories. For example, the area covered by a department in a country can first be seen as a set composed of '*venues*' scattered in the considered area. In this example, the datatype is $T =$ '*venues*'. But each venue can itself be precisely identified as a football field, a pharmacy, a library, or a spectacle venue. In this context, the main datatype is still $T =$ '*venues*', but here $T$ is composed of the $n = 4$ following subtypes: $T_1 =$ '*football fields*', $T_2 =$ '*pharmacies*', $T_3 =$ '*libraries*' and $T_4 =$ '*spectacle venues*'. Consequently, a department can also be viewed as a set composed of elements whose possible subtypes are '*football fields*', '*pharmacies*', '*libraries*', and '*spectacle venues*'.

In order to distinguish these two categories in the rest of this paper, the expression '*dataset*' will refer to the first category, while the expression '*data collection*' will designate the second category.

The result of a comparison between two datasets $X$ and $Y$ can most often be a logic outcome that can be evaluated as *true* or *false* (e.g: is $X$ a subset of $Y$?). As analyzed in [3][4], such binary spatial relations can be classified according to the three following geometric categories: *topological* (e.g: does $X$ touches $Y$?), *projective* (e.g: is $X$ inside the concavity of $Y$?) and *metric* (e.g: is $X$ bigger than $Y$?). But there are so many situations where it is hardly possible to perform such comparisons, this is for example the case for the datasets $X = \{1, 2, 4, 6\}$ and $Y = \{1, 3, 4, 5\}$. In such situations and for the particular case of metric spaces where a metric is defined, the comparison can still be realized and the result of this comparison is generally a positive numeric value representing the formal distance (as defined in Eq. 1) between the two metric spaces. It is in this perspective that many metrics have been proposed to address the problem of comparing two datasets [4–10]. These existing measures can be organized into two groups:

1. In the first group, *cluster-based* comparison is performed [4, 5]. Here, existing distance measures between two elements are used in a specific algorithm for deriving the result.
2. In the second group, *model-based* comparison is performed [6–10]. A model is initially designed for capturing the content of each dataset. The comparison between the two datasets is then realized through the comparison of their associated models. These approaches are more reliable than *cluster-based* approaches because they consider each dataset as a single entity.

The aforelisted existing techniques share at least two common limitations, irrespective of the group to which they belong. On the one hand, they do not consider the way how elements are scattered inside the datasets. However, the dispersion of the elements inside a dataset is obviously a fundamental descriptor for this dataset. On the other hand, none of these existing techniques specifically addresses the problem of comparing two data collections.

This paper attempts to overcome these drawbacks through the proposal of a histogram-based technique for comparing two metric spaces using hidden Markov models (HMMs). This is realized by initially transforming each element $x$ of a dataset $X$ into a histogram which gradually captures the data dispersion inside the neighborhood of $x$. Proceeding this way, each dataset $X$ becomes a set $H_X$ containing the $|X|$ histograms derived from its elements. The HMM-based

[3] See the first footnote of [2] at the end of page 3.

[4] See Table 1 (column 2) of [3].

technique proposed in [8] is then used for associating a HMM $\lambda_X$ with $X$ such that $\lambda_X$ learns the bin values and the visual shapes of the histograms found in $H_X$. Metadata derived from $\lambda_X$ are later saved as the components of a descriptor vector $\vec{X}$ associated with $X$. Finally, the comparison between two metric spaces is performed through the comparison of their respective associated descriptor vectors using existing distance or similarity measures between two vectors. The proposed approach is experimented for the comparison of datasets and data collections containing geolocations or stars in the celestial sphere.

The rest of this paper is organized as follows: the state of the art is presented in Sect. 2, followed by a summarized presentation of HMMs in Sect. 3. The description of the proposed approach is realized in Sect. 4, while experimental results are presented in Sect. 5. In Sect. 6, a discussion related to relevant research problems and perspectives that can be explored in future work is proposed. The last section is dedicated to the Conclusion.

## 2 State of the art

### 2.1 Comparison of two single elements

Many existing metrics used for comparing two datasets are based on the comparison of two single elements belonging to the same datatype. For this reason, a short overview of some relevant metrics used for comparing two single elements is first proposed in this section. Considering both, the high number of the possible existing datatypes and the high number of existing metrics, it is not reasonable to make an exhaustive review of all the existing metrics in this paper. Nevertheless, a review of relevant existing metrics between two elements belonging to one of the five following datatypes is performed in the next sections: $\mathbb{R}^n$, histograms, trees, geolocations on the earth and stars in the celestial sphere.

#### 2.1.1 Comparison of two elements of $\mathbb{R}^n$

An element of $\mathbb{R}^n$ is an object (vector, point) which is fully described by its $n$ components, all belonging to $\mathbb{R}$. They are widely used in several domains including mathematics, physics and computer science. The most used distance for comparing two elements of $\mathbb{R}^n$ is the *Euclidean distance*. Given two elements $x$ and $y$ of $\mathbb{R}^n$, this measure evaluates the straight-line distance between $x$ and $y$. Another popular distance in $\mathbb{R}^n$ is the *Manhattan distance* which evaluates the distance traveled by a taxi when it moves from $x$ to $y$ in a city where the streets are organized as a grid. Equation 3 shows how to compute these distances for two elements $x = [x_1, \ldots, x_n]$ and $y = [y_1, \ldots, y_n]$ of $\mathbb{R}^n$. Formal details about more than 60 other existing distances are available in [11][5].

$$\begin{aligned} d_1(x,y) &= \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (\textit{Euclidean}) \\ d_2(x,y) &= \sum_{i=1}^{n}|x_i - y_i| \qquad (\textit{Manhattan}) \end{aligned} \tag{3}$$

Many similarities are also available for comparing two elements of $\mathbb{R}^n$. This is the case of the *Cosine similarity* which is the cosine of the angle $\widehat{(x,y)}$ between $x$ and $y$. This angle is obtained by computing the arccos of the *Cosine similarity*. If the comparison aims at discovering any rank correspondence between $x$ and $y$, several correlation coefficients are available and among them we can list: the *Pearson correlation coefficient*, the *Spearman's $\rho$ rank coefficient*, the *Kendall $\tau$ coefficient* and the *Goodman-Kruskal's $\gamma$ rank correlation*. A detailed presentation of these correlation coefficients is available in [12][6].

#### 2.1.2 Comparison of two histograms

A histogram can be seen as a finite ordered sequence of bin values (positive real numbers) whose variations are meaningful because they determine the visual shape of the histogram [8][7]. Histogram-based local descriptors are used in several application domains including music, image and text processing. Most of the existing measures used for comparing two histograms $h_1$ and $h_2$ apply bin-to-bin functions. It is for example the case of the *Intersection distance* which returns the sum of minimum bin-to-bin values between $h_1$ and $h_2$ [13]. It is also the case of the *$\chi^2$-statistics divergence* which evaluates how likely it is that any observed difference between $h_1$ and $h_2$ arose by chance [14].

Unlike the aforelisted metrics, other metrics using cross-bin information for comparing two histograms have also been proposed. This is the case of the *Diffusion distance* [15] which defines the distance between two histograms as a temperature field and considers the diffusion process on the field. Another important member of this category of metrics is the *Earth Movers distance* [16] which evaluates the least amount of work needed to transport earth or mass from $h_1$ to $h_2$. Other authors rather focused on metric learning approaches. In [17] for example, a non-linear metric learning approach dedicated to histograms is proposed. This metric generalizes the bin-to-bin *$\chi^2$-statistics distance* in order to learn a metric that strictly preserves the histograms

[5] See Tables 1, 2, 3, 4, 5, 6, 7, 8 and Table 10 of [11].

[6] See Section 2.5 of [12].

[7] See the Introduction of [8].

properties. This approach exhibits more reliable results than the original $\chi^2$-*statistics distance*. A detailed state of the art on existing metrics between two histograms is available in [8][8].

#### 2.1.3 Comparison of two trees

A tree is a connected acyclic graph. No specific details related to graph/tree concepts will be presented in this paper. Nevertheless, a detailed overview on graph theory and applications is available in [18]. Many distances have yet been proposed for comparing trees and they are mostly based on one of the five following principles:

1. *Tree edit* [19–21].
2. *Tree alignment* [22, 23].
3. *Tree inclusion* [24, 25].
4. *Tree pattern matching* [26–28].
5. *Subtrees or supertrees similarity* [29–31].

The *tree edit* distance is based on the analysis of the number of edit operations required to transform a tree $t_1$ into another tree $t_2$. The three following edit operations are considered for a given node: insertion, deletion and substitution. *Tree alignment* is a particular case of *tree edit* where insertions are always performed before deletions. Let $t_1$ and $t_2$ be two rooted trees with labeled nodes. $t_1$ is *included* in $t_2$ if there is a sequence $sec(t_1, t_2)$ of node deletions performed on $t_2$ which makes $t_2$ isomorphic to $t_1$. The tree inclusion problem is to decide if $t_1$ can be included in $t_2$ and the *tree inclusion* distance is the sum of the costs of the delete operations found in $sec(t_1, t_2)$. *Tree pattern matching* consists in finding the instances of a given pattern tree in a specific target tree [26–28]. *Subtrees or supertrees similarity* are generally realized by finding the maximum agreement sub-tree [29], the largest common sub-tree [30] or the smallest common super-tree [31]. Details related to all these principles are available in [10][9].

#### 2.1.4 Comparison of two geolocations on Earth

The acronym *GPS* stands for *Global Positioning System*. It is a system used for worldwide navigation and surveying. The system can localize any element (person, venue, object) anywhere on the Earth's surface. The expression '*GPS coordinates*' generally refers to a pair of coordinates: the *latitude* and the *longitude*. In the experiments realized in this paper, the *'Decimal degrees'* notation of these coordinates is adopted. In this notation, the latitude and the longitude are expressed as decimals belonging to the interval $[-180, 180]$. Consider two elements $A$ and $B$ whose *GPS* coordinates are respectively $(lat_A, lon_A)$ and $(lat_B, lon_B)$. The *Haversine formula* [32] presented in Eq. 4 is selected in this paper for calculating the distance between $A$ and $B$. In this equation, the function $rd(\varepsilon)$ returns the value in radians of the angle $\varepsilon$ initially measured in degrees. The result of Eq. 4 is in Kilometers and the constant 6371 is the mean radius of the Earth in Kilometers.

$$\begin{aligned} d_{gps}(A,B) &= 6371\times\arccos(c_1 + c_2) \\ \text{where } c_1 &= \sin(rd(lat_A))\times\sin(rd(lat_B)) \\ \text{and } c_2 &= \cos(rd(lat_A))\times\cos(rd(lat_B))\times\cos(c_3) \\ \text{and } c_3 &= rd(lon_A - lon_B) \end{aligned} \tag{4}$$

#### 2.1.5 Comparison of two stars

In order to localize a star in the celestial sphere, one must select a planet from which this star is observed. Generally, the Earth is the selected planet. A star observed from the Earth can be fully localized in the celestial sphere by its *celestial coordinates* (*Ra*, *Dec*) and the distance $r$ between the Earth and the considered star. *Ra* is the *right ascension* of the star, generally expressed in degrees between $-90°$ and $+90°$. *Dec* is the *declination* of the star, generally expressed in hours (between $0h$ and $24h$), but it can be converted in degrees in which case 360° corresponds to $24h$. Consider two stars $S_1$ and $S_2$ observed from the Earth whose celestial coordinates are respectively $(a_1, b_1)$ and $(a_2, b_2)$. The values $a_1, a_2$ and $b_1, b_2$ are respectively the right ascensions and the declinations of the considered stars. Let us now assume that the distance from the Earth to $S_1$ is $r_1$ and the distance from the Earth to $S_2$ is $r_2$. It is demonstrated in [33][10] that the distance $d_{star}$ between $S_1$ and $S_2$ is computed by Eq. 5 where the right ascensions are converted in degrees. This distance measure has been selected in this paper to be the distance between two stars. The unit of the result of $d_{star}$ is the light-year.

$$\begin{aligned} d_{star}(S_1,S_2) &= \sqrt{r_1^2 + r_2^2 - 2.r_1.r_2.\cos\gamma} \quad \text{where} \\ \cos\gamma &= \sin b_1.\sin b_2 + \cos b_1.\cos b_2.\cos(a_1 - a_2) \end{aligned} \tag{5}$$

[8] See Sect. 2.1 and Table 1 of [8].
[9] See Section 2.2 of [10].
[10] See Eqs. 1 and 2 of [33].

## 2.2 Existing metrics for comparing two datasets

### 2.2.1 Cluster-based comparison

Consider two datasets $X$ and $Y$, both containing elements belonging to the same datatype $T$ which can be $\mathbb{R}^n$, histograms, trees, geolocations, stars, etc. Let *dist* be a metric between any two elements of type $T$, the measures reviewed in Sect. 2.1 can be used for this purpose. The following cluster distances analyzed in [4] and whose formulas are presented in Eq. 6 are most often used for comparing $X$ and $Y$:

1. The *minimum linkage* distance also called the *single-link* distance: Here, the distance between the two nearest elements of $X$ and $Y$ is considered.
2. The *maximum linkage* distance also called the *complete-link* distance: this measure evaluates the distance between the most distant elements of $X$ and $Y$.
3. The *average linkage* distance which is the mean of the distances between every pair $(x, y) \in X \times Y$.
4. The *centroid linkage* distance where an initial centroid calculation is realized in each dataset. If we respectively note $\tilde{X}$ and $\tilde{Y}$ the centroids of $X$ and $Y$, then the distance between $X$ and $Y$ is computed as the distance between $\tilde{X}$ and $\tilde{Y}$.

$$\begin{aligned} Minimum(X,Y) &= \min_{(x,y)\in X\times Y} \{dist(x,y)\} \\ Maximum(X,Y) &= \max_{(x,y)\in X\times Y} \{dist(x,y)\} \\ Average(X,Y) &= \frac{1}{|X|.|Y|}\sum_{x\in X}\sum_{y\in Y} dist(x,y) \\ Centroid(X,Y) &= dist(\tilde{X},\tilde{Y}) \end{aligned} \tag{6}$$

Beside these metrics, it is also possible to compute the distance between $X$ and $Y$ when they are members of a particular taxonomy. A well-known method to do this is the *Ward's method* which computes the *merging cost* $\Delta(X, Y)$ of combining $X$ and $Y$ [5]. Equation 7 shows how to compute the merging cost of $X$ and $Y$.

$$\Delta(X,Y) = \left(\frac{|X|.|Y|}{|X|+|Y|}\right) \times dist(\tilde{X},\tilde{Y})^2 \tag{7}$$

### 2.2.2 Model-based comparison

When a formal model is designed for each dataset, it becomes possible to compare two datasets through the comparison of their associated models.

In 2007, a *Malahanobis* distance between datasets based on summary statistics is defined [6]. Here, datasets are finite collections of samples belonging to the same sample space $\beta$ (itemsets, binary data, etc.). Before comparing two datasets $D_1$ and $D_2$, a *feature function* $\kappa : \beta \rightarrow \mathbb{R}^n$ which maps every element of $\beta$ to a vector in $\mathbb{R}^n$ is initially defined. The authors then model each dataset $D_i$ as its *frequency* $f_i = \frac{1}{|D_i|}\sum_{\omega\in D_i}\kappa(\omega)$ which is the average of the values of $\kappa$ taken over $D_i$. Finally, Theorem 1 of [6] shows how to compute the distance $d_{CM}(D_1, D_2|\kappa)$ between two datasets $D_1$ and $D_2$ considering the feature function $\kappa$ by comparing their frequencies (models) $f_1$ and $f_2$. Consequently, the efficiency of $d_{CM}$ strongly depends on the choice of $\kappa$.

In 2017, the *Gromov-Hausdorff distance* is proposed for comparing two compact metric spaces [7]. More formally, in order to compare two compact metric spaces $X$ and $Y$, one initially needs to consider a third sufficiently rich compact metric space $Z$ inside which isometric copies $\phi_X(X)$ and $\phi_Y(Y)$ of $X$ and $Y$ are respectively available. $\phi_X(X)$ and $\phi_Y(Y)$ are considered here as the models respectively associated with $X$ and $Y$. Equation (3.1) of [7] shows how to finally compute the *Gromov-Hausdorff distance* between $X$ and $Y$ by computing the simple *Hausdorff distance* between $\phi_X(X)$ and $\phi_Y(Y)$ which roughly evaluates the inclusion of each dataset in specific subsets of the other.

In 2018, an accurate HMMs-based approach for comparing finite sets of histograms was proposed [8]. Consider $N$ datasets $H_1, \ldots, H_N$ containing histograms. In that work, each histogram was first represented as a Markov chain. Then, a HMM $\lambda_i$ was trained to capture the visual shapes and the bin values of the $|H_i|$ histograms in $H_i$ $(1 \leq i \leq N)$. Finally, the similarity rate between two distinct datasets $H_i$ and $H_j$ $(1 \leq i, j \leq N)$ was obtained by considering the value of the similarity between their respective associated models $\lambda_i$ and $\lambda_j$, weighted by a suitable amplitude coefficient. The technique proposed in [8] exhibited better results than existing techniques in several domains including color images comparison, text documents comparison, automatic taxonomy generation of music genres and the comparison of the curves associated with 2D and 3D functions.

In 2020, an efficient generic approach for automatic taxonomy generation using HMMs was proposed [9]. Consider $N$ classes (datasets) $C_1, \ldots, C_N$ containing elements belonging to the datatype $T$. In that work, an analysis of the neighborhood of each element is first performed. More precisely, the number of elements of each class $C_i$ found in this neighborhood is captured inside a Markov chain from the most dominant class to the least dominant. The Markov chains resulting from this analysis are used to initialize, then to train a HMM for each class. The similarities between any two classes $C_i$ and $C_j$ $(1 \leq i, j \leq N)$, derived from the similarity between their respective associated HMMs, are finally used for constructing the taxonomy.

The taxonomies resulting from this technique outperformed the state of the art in the hierarchical classification of 20 widespread datasets containing elements belonging to several datatypes.

More recently in 2021, a customizable HMM-based technique was proposed for accurately comparing two tree sets, irrespective of the characteristics verified by the trees they contain [10]. This technique was therefore applicable to rooted and unrooted trees, to labeled and unlabeled trees, as well as to weighted and unweighted trees. Unlike former techniques for tree comparison, this technique enables the user to explicitly specify the node properties on which the comparison must be performed. More formally, a HMM was associated with each tree set for each target node property. The model associated with a tree set $T$ for the target node property $p$ learned how much the nodes of the trees in $T$ verify property $p$. The resulting HMMs were finally compared for deriving both, a distance and a similarity between the two finite tree sets. Flat classification experiments conducted on two online available synthetic datasets[11], both containing 4 classes of 100 rooted ordered trees whose node properties were clearly defined, exhibited a perfect accuracy of 100% for these two datasets when the tree distance proposed in [10] was selected as metric for the *Nearest Neighbor* classifier. This performance is 41% higher than the accuracy exhibited when the *tree Edit* distance is used. Table 1 summarizes the main characteristics of the existing techniques for comparing two datasets reviewed in this section.

### 2.3 Problem statement

Existing *cluster-based* techniques [4, 5] and *model-based* techniques [6–10] for comparing datasets do not take into consideration the dispersion of the elements inside each dataset. However, this is obviously a fundamental descriptor of each dataset for several real-world applications. Furthermore, comparing two data collections remains an open field of research because existing techniques do not specifically address this problem. Consider for example an employee of a company that must move from a city $X$ to a city $Y$ of his choice for professional reasons. If that employee is very attached to the way in which the venues (football fields, libraries, pharmacies, spectacle venues, etc.) are scattered in $X$, he will therefore seek to settle in a city $Y$ whose venues are scattered in a similar way. But, he will do the opposite in the event that the dispersion of venues in $X$ does not suit him. In such a context, a metric that can evaluate the distance between $X$ and $Y$ while considering the dispersion of venues in these two cities is required.

This paper aims at solving these limitations through the proposal of a new histograms-based approach for comparing two metric spaces using HMMs. This is realized by transforming a metric space $X$ into a set $H_X$ of histograms which gradually capture the data dispersion inside the neighborhood of each element of $X$. Then, following [8], a HMM $\lambda_X$ is trained to learn the bin values and the visual shapes of the histograms contained in $H_X$. Meta-data derived from $\lambda_X$ are later saved as the components of a descriptor vector $\vec{X}$ associated with $X$. The comparison between two metric spaces is finally performed through the comparison of their respective associated descriptor vectors using existing distance or similarity measures between two vectors.

## 3 Summarized presentation of HMMs

The current section is dedicated to a summarized presentation of some essential HMMs-related concepts. More detailed presentations can be found in [8–10] as well as in the tutorial on HMMs and selected applications in speech recognition available in [34].

A HMM $\lambda = (S, \vartheta, A, B, \pi)$ is fully characterized by [34][12]:

1. Its set $S = \{s_1, \ldots, s_N\}$ of states.
2. Its set $\vartheta = \{v_1, \ldots, v_M\}$ of symbols.
3. Its state transition probability matrix $A$ such that $A[s_i, s_j]$ is the probability that $\lambda$ transits from state $s_i$ to state $s_j$.
4. Its symbols probabilities matrix $B$ such that $B[s_i, v_k]$ is the probability that $\lambda$ observes symbol $v_k$ from state $s_i$.
5. Its initial state probability vector $\pi$ such that $\pi[s_i]$ is the probability that the initial state of $\lambda$ is $s_i$.

For convenience, the notation $\lambda = (S, \vartheta, A, B, \pi)$ is generally reduced to $\lambda = (A, B, \pi)$. A HMM can be used for generating a *Markov chain* which is a finite sequence of states such that a specific symbol is observed from each state of the sequence.

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 $\lambda$ is efficiently computed by the *Forward-Backward* algorithm [34][13] which runs in $\theta(T.N^2)$.

The *Baum-Welch* algorithm [34][14] is generally used for iteratively re-estimating the parameters of a HMM $\lambda = (A, B, \pi)$ in order to maximize the value of $Prob(O|\overline{\lambda})$, where $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ is the re-estimated model. This algorithm runs in $\theta(\gamma.T.N^2)$ where $\gamma$ is the user-defined maximum number of iterations. The *Baum-Welch* algorithm can also be used to train a HMM for multiple sequences [15]. The

[11] http://perso-etis.ensea.fr/sylvain.iloga/FirstLast/index.html.

[12] See Section II-B of [34].

[13] See Section III-A of [34].

[14] See Section III-C of [34].

[15] See Section V-B of [34].

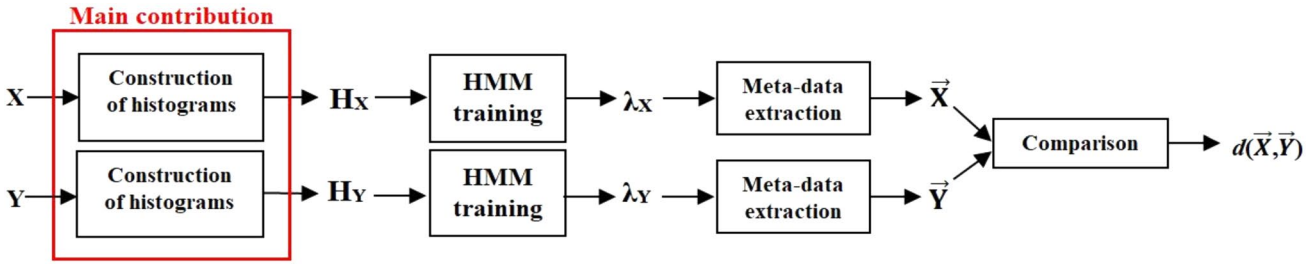



**Fig. 1** Proposed methodology for comparing two metric spaces $X$ and $Y$

algorithm maximizes the value of $Prob(O|\overline{\lambda}) = \sum_{k=1}^{K} Prob(O^{(k)}|\overline{\lambda})$ 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 $O$. The time complexity of the *Baum-Welch* algorithm for multiple sequences is approximated by $\theta\left(\gamma.(\sum_{k=1}^{K} T_k).N^2\right)$

A stationary distribution of a HMM $\lambda = (A, B, \pi)$ is a vector $\varphi = (\varphi[s_1], \ldots, \varphi[s_N])$ such that $\varphi[s_j]$ estimates the overall proportion of time spent by $\lambda$ in state $s_j$ after a sufficiently long time [35][16]. $\varphi$ can be extracted from any line of the matrix $A^r = A \times A \times \ldots \times A$ ($r$ times) when $r \to +\infty$. Therefore, the computation of $\varphi$ requires $\theta(r.N^3)$ arithmetic operations.

## 4 The proposed approach

### 4.1 Main idea

Consider a finite metric space $X$ containing elements from a datatype $T$ and let *dist* be a metric between any two elements of $T$. Given a user-defined positive real number $R$, the main idea of the technique proposed in the current paper is that each individual $x \in X$ can be characterized by the dispersion of the individuals localized in its neighborhood of radius $R$. This idea arises from a limitation of the main idea of [9][17] where the neighborhood of each instance was also analyzed, but the resulting analysis suffered from the lack of information related to the dispersion of the instances inside the neighborhood[18].

To achieve this goal, we first divide the neighborhood of $x$ into $p$ consecutive slices of identical width, where $p$ is a user-defined integer. This division allows us to graduate the neighboring zones of $x$ depending on their proximity to $x$. As an example, for $p = 5$, we can adopt the following graduation: '*very close*', '*close*', '*slightly distant*', '*distant*' and '*very distant*'. Thereafter, the number of neighbors found in each zone is saved into a *characteristic histogram* $h_x$ in such a way that the $i^{th}$ bin value $h_x(i)$ is the number of neighbors found in the $i^{th}$ slice ($1 \leq i \leq p$). When this principle is applied for each individual $x \in X$, a dataset $H_X$ containing $|X|$ characteristic histograms is obtained.

### 4.2 Methodology

The proposed methodology for comparing two metric spaces presented in Fig. 1 clearly specifies the main contribution of the current paper. As it can be observed in that figure, the proposed technique is composed of the 4 following steps:

1. The transformation of each metric space into a set of characteristic histograms by performing a gradual analysis of the neighborhood of every individual in the metric space.
2. The bin values and the visual shapes of the histograms contained in each set of characteristic histograms are learned through the training of a HMM using the *Baum-Welch* algorithm following [8].

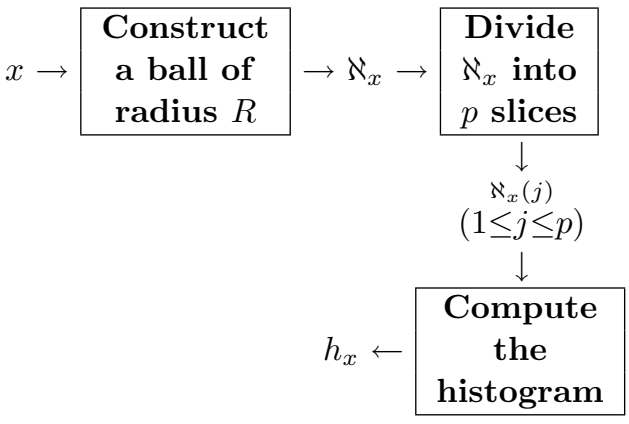
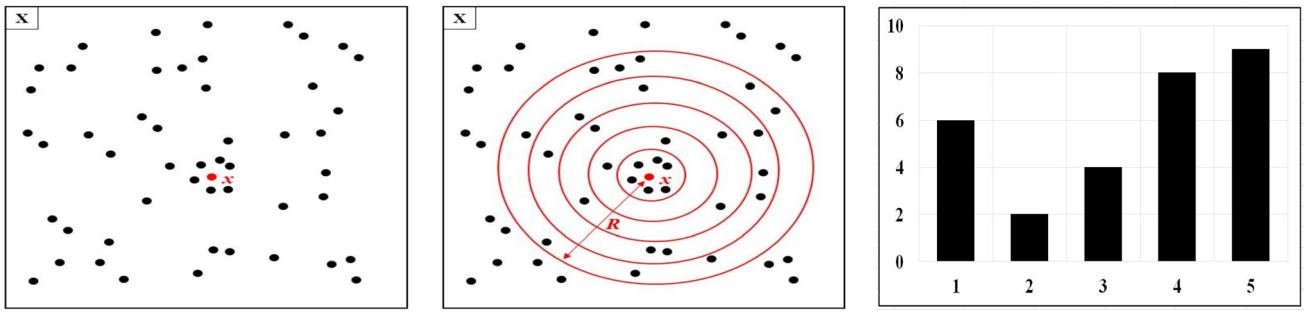



**Fig. 2** Construction of the characteristic histogram $h_x$ of an instance $x$ in a dataset

[16] See Definition 9.1 of [35].

[17] See Section 4.1 of [9].

[18] See the Conclusion of [9].




(a) Identification of $x$ in $X$ (b) Division of $\aleph_x$ into 5 slices (c) The characteristic histogram $h_x$

**Fig. 3** Construction of the characteristic histogram $h_x$ of the instance $x \in X$

3. Analogically to [36], meta-data extracted from the resulting HMMs are saved as the components of a descriptor vector for each metric space. These meta-data are related to the overall behavior of each HMM regarding every symbol after a sufficiently long time.
4. The comparison between the two input metric spaces is performed through the comparison between their associated descriptor vectors using any existing distance or similarity $d$ between vectors.

## 4.3 Construction of the histograms

### 4.3.1 The case of datasets

Consider a finite dataset $X$ containing elements from a datatype $T$ and let *dist* be a metric between any two elements of $T$. Consider now two user-defined numbers $R$ and $p$ respectively representing the radius of the neighborhood of each instance and the number slices resulting from the division of this neighborhood. As it is summarized in Fig. 2, the characteristic histogram $h_x$ derived from each $x \in X$ is constructed as follows:

1. Construct a ball of radius $R$ centered at $x$ named $\aleph_x$ which materializes the neighborhood of $x$ in $X$.
2. Divide $\aleph_x$ into $p$ consecutive slices such that $\aleph_x(j)$ refers to the elements of $X$ found inside the $j^{th}$ slice, each slice having a width of $\frac{R}{p}$. Equation 8 gives the formal definition of $\aleph_x(j)$.
3. For every $(1 \leq j \leq p)$, compute $h_x$ using Eq. 9.

$$\aleph_x(j) = \left\{ y \in X | (j-1).\frac{R}{p} \leq dist(x,y) < j.\frac{R}{p} \right\} \tag{8}$$

$$h_x(j) = |\aleph_x(j)| \tag{9}$$

When this principle is applied to each instance of $X$, the set $H_X = \{h_x | x \in X\}$ is obtained. As an example, consider the instance $x$ of the dataset $X$ identified in Fig. 3a, where all the instances in $X$ are represented as dots. In b, the neighborhood $\aleph_x$ of $x$ having as radius $R$ is divided into $p = 5$ consecutive slices of identical width represented by circles. Finally, the characteristic histogram $h_x$ of $x$ is constructed in Fig. 3c in such a way that $h_x(j)$ is the number of neighbors found in the $j^{th}$ slice $(1 \leq j \leq 5)$.

### 4.3.2 The case of data collections

As described in Fig. 4, given a data collection $A = \{A_1, \ldots, A_n\}$, the following procedure describes how to construct the characteristic histogram $h_x$ associated with any instance $x \in A$, irrespective of its corresponding subtype:

1. Construct a ball of radius $R$ centered at $x$ named $\aleph_x$ which materializes the neighborhood of $x$ in $A$.
2. Divide $\aleph_x$ into $p$ consecutive slices such that $\aleph_x(j)$ refers to the elements of $A$ found inside the $j^{th}$ slice irrespective of their corresponding subtypes, each slice having a

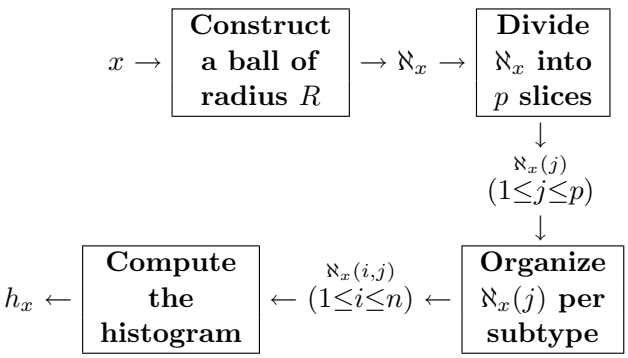
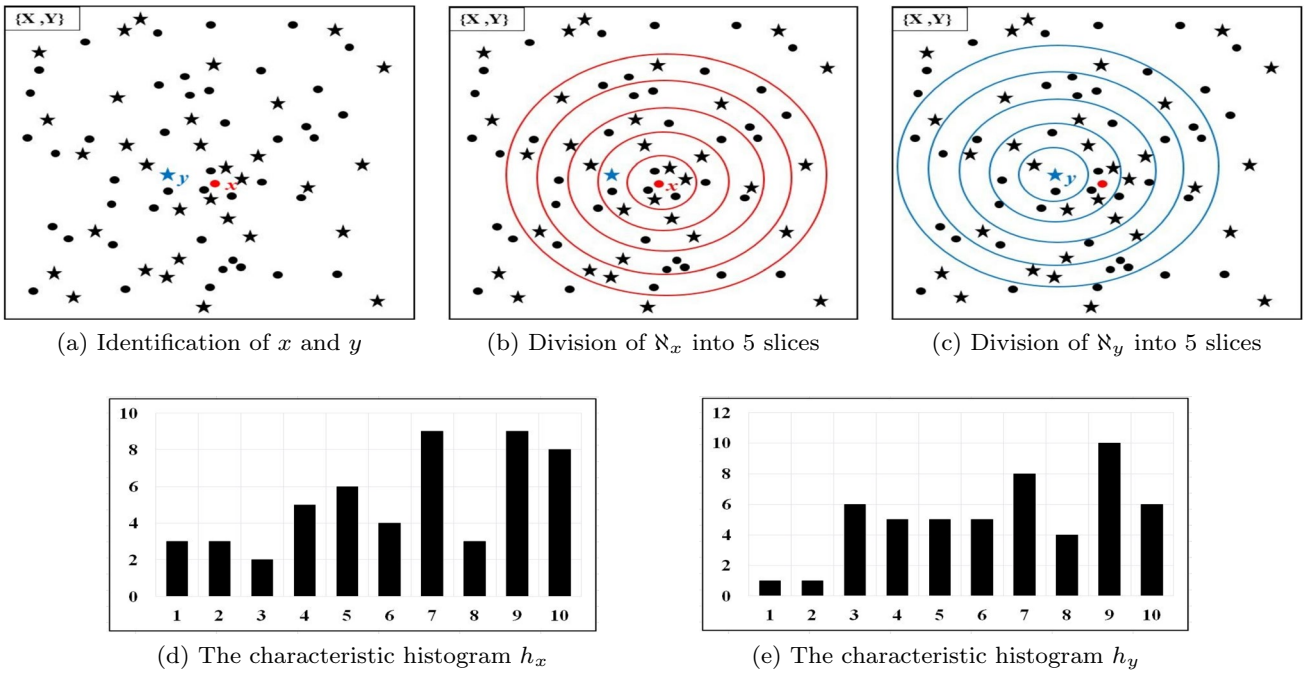



**Fig. 4** Construction of the characteristic histogram $h_x$ of an instance $x$ in a data collection




(a) Identification of $x$ and $y$ (b) Division of $\aleph_x$ into 5 slices (c) Division of $\aleph_y$ into 5 slices

(d) The characteristic histogram $h_x$ (e) The characteristic histogram $h_y$

**Fig. 5** Construction of the characteristic histograms $h_x$ and $h_y$ associated with the instances $x \in X$ and $y \in Y$

width of $\frac{R}{p}$. Equation 10 gives the formal definition of $\aleph_x(j)$.

3. For every $(1 \leq j \leq p)$, organize the content of $\aleph_x(j)$ regarding the subtypes of its elements such that $\aleph_x(i,j)$ refers to the elements of $\aleph_x(j)$ belonging to $A_i$ as shown in Eq. 11.
4. For every $(1 \leq i \leq n)$ and every $(1 \leq j \leq p)$, compute $h_x$ using Eq. 12.

$$\aleph_x(j) = \left\{ y \in A | (j-1).\frac{R}{p} \leq dist(x,y) < j.\frac{R}{p} \right\} \tag{10}$$

$$\aleph_x(i,j) = \left\{ y \in \aleph_x(j) | y \in A_i \right\} \tag{11}$$

$$h_x[n.(j-1)+i] = |\aleph_x(i,j)| \tag{12}$$

Consider for example a data collection $\beta = \{X, Y\}$ composed of two subsets: $X$ containing dots and $Y$ containing stars, both dots and stars being subtypes of a datatype $T$. Consider now the instances $x \in X$ and $y \in Y$ identified in Fig. 5a. The neighborhoods $\aleph_x$ and $\aleph_y$ of $x$ and $y$ are respectively divided into $p = 5$ slices represented by circles as shown in Fig. 5b, c. When Eq. 11 is applied, the characteristic histograms $h_x$ and $h_y$ respectively presented in Fig. 5d, e are derived. When this procedure is applied on a data collection $A = \{A_1, \ldots, A_n\}$, the collection $H_A = \{H_{A_1}, \ldots, H_{A_n}\}$ containing $n$ sets of characteristic histograms is obtained.

### 4.4 HMM training

In order to derive the HMM associated with each set of characteristic histograms, the authors of [8] initially normalize the histograms of each input set in order to make their bin values belong to the interval [0, 100]. Given a set $H_X$ of characteristic histograms, this normalization first consists in dividing each bin value by the highest bin value appearing in $H_X$ (noted here as $\overline{X}$), all the resulting bin values are then multiplied by 100. Thereafter, each normalized histogram is transformed into a Markov chain where each state is the absolute value of the difference between two consecutive bin values, while each bin value is considered as a symbol. The resulting Markov chains are finally used to initialize then to train a HMM with the *Baum-Welch* algorithm for multiple sequences.

### 4.5 Meta-data extraction

#### 4.5.1 Vector associated with a dataset

The descriptor vector $\vec{X} = [X^1, X^2, \ldots, X^{|\vartheta|+1}]$ associated with a dataset $X$ is derived from its corresponding HMM $\lambda_X = (A_X, B_X, \pi_X)$ by analyzing its behavior regarding each symbol analogically to [36][19] where the authors realized human activity recognition using HMMs. More precisely, we consider that:

[19] See Section IV-E and Eq. 7 of [36].

1. The $k^{th}$ component $X^k$ of $\vec{X}$ $(1 \leq k \leq |\vartheta|)$ is the overall proportion of time spent by $\lambda_X$ observing symbol $v_k$ after a sufficiently long time, irrespective of the state from which this symbol is observed.
2. The last component $X^{|\vartheta|+1}$ of $\vec{X}$ is the value $\overline{X}$ used during the normalization of the input histograms (to consider the effects of this normalization).

In order to compute the value of $X^k$ $(1 \leq k \leq |\vartheta|)$, the overall proportion of time spent by $\lambda_X$ observing symbol $v_k$ from each state $s_i$ after a sufficiently long time is evaluated as follows:

1. The overall proportion of time spent by $\lambda_X$ in state $s_i$ after a sufficiently long time is evaluated. This proportion is the $i^{th}$ component $\varphi_X[s_i]$ of the stationary distribution of $\lambda_X$.
2. The value obtained at step 1 is then multiplied by the probability of observing $v_k$ from state $s_i$ which is $B_X[s_i, v_k]$.

The value of $X^k$ is finally obtained by repeating this process for every state $s_i$ and summing the resulting proportions as shown in Eq. 13.

$$\begin{cases} \vec{X} = [X^1, X^2, \ldots, X^{|\vartheta|}, \overline{X}] & \text{where} \\ X^k = \sum_{i=1}^{N} \left( \varphi_X[s_i] \times B_X[s_i, v_k] \right) & \text{with } (1 \leq k \leq |\vartheta|) \end{cases} \tag{13}$$

#### 4.5.2 Vector associated with a data collection

We propose to consider that the descriptor vector $\vec{A}$ associated with a data collection $A = \{A_1, \ldots, A_n\}$ is obtained by realizing the sequential concatenation of the components of the descriptor vectors $\overrightarrow{A_1}, \ldots, \overrightarrow{A_n}$ respectively associated with $A_1, \ldots, A_n$ as shown in Eq. 14.

$$\vec{A} = [\underbrace{A_1^1, A_1^2, \ldots, A_1^{|\vartheta|}, \overline{A}_1}_{\overrightarrow{A_1}}, \ldots, \underbrace{A_n^1, A_n^2, \ldots, A_n^{|\vartheta|}, \overline{A}_n}_{\overrightarrow{A_n}}] \tag{14}$$

### 4.6 Comparison of two metric spaces

Consider two metric spaces $X$ and $Y$. For any existing distance (resp. similarity) measure $d$ between two vectors, we define in Eq. 15 the corresponding *$\sigma$-distance* (resp. *$\sigma$-similarity*) between $X$ and $Y$ noted $\sigma^d$, as the distance (resp. similarity) $d$ between their respective associated descriptor vectors $\vec{X}$ and $\vec{Y}$.

$$\sigma^d(X, Y) = d(\vec{X}, \vec{Y}) \tag{15}$$

### 4.7 Pseudo-code of the proposed technique

As it is described in Algorithm 1, the proposed method for comparing two metric spaces $X$ and $Y$ according to a selected metric $d$ between vectors simply consists in first generating their corresponding descriptor vectors calling Algorithm 2 (lines 1-2) before performing the comparison between these two vectors using $d$ following the principle stated in Section 4.3 (line 3). The parameters $p$ and $R$ of Algorithms 1 and 2 respectively refer to the number of slices and the radius of the neighborhood of each instance which is analyzed.

**Algorithm 1** *Comparison*

**Inputs:** $X, Y, p, R, d$
**Output:** $z$

1: $\vec{X} \leftarrow DescriptorVector(X, p, R)$
2: $\vec{Y} \leftarrow DescriptorVector(Y, p, R)$
3: $z \leftarrow d(\vec{X}, \vec{Y})$
4: **return** $z$

Algorithm 2 provides a step-by-step description of the derivation of the descriptor vector $\vec{X}$ associated with a metric space $X$. After initializing the set $H_X$ of characteristic histograms to the emptyset (line 1), $H_X$ is gradually constructed by adding the characteristic histogram $h_x$ of each instance $(x \in X)$ following the procedure described in Sect. 4.3 (lines 2-5). The HMM $\lambda_X$ associated with $X$ is then trained using the content of $H_X$ as explained in Sect. 4.4 (line 6). Finally, meta-data extracted from $\lambda_X$ enable to derive and to return the descriptor vector $\vec{X}$ of $X$ following the principle presented in Sect. 4.5 (lines 7-8).

**Algorithm 2** *DescriptorVector*

**Inputs:** $X, p, R$
**Output:** $\vec{X}$

1: $H_X \leftarrow \emptyset$
2: **for each** $(x \in X)$ **do**
3: $\quad h_x \leftarrow HistogramConstruction(x, p, R)$
4: $\quad H_X \leftarrow H_X \cup \{h_x\}$
5: **end for**
6: $\lambda_X \leftarrow HmmTraining(H_X)$
7: $\vec{X} \leftarrow MetaDataExtraction(\lambda_X)$
8: **return** $\vec{X}$

**Table 2** Number of communes in the 4 selected prefectures in France in 2014

| | Cergy | Evry | Melun | Versa. |
|---|---|---|---|---|
| Communes | 186 | 183 | 478 | 250 |

### 4.8 Accuracy and efficiency of the proposed technique

The technique proposed in the current paper for comparing two metric spaces:

1. Is based on a gradual analysis of the neighborhood of each instance, which is a refinement of the neighborhood's analysis performed in [9] where the dispersion of the instances inside the neighborhood was not taken into account. The accuracy and the efficiency of the neighborhood's analysis realized in [9] have been demonstrated through the good performances exhibited by that work.
2. Relies on the HMMs generated in [8] whose accuracy and efficiency were also demonstrated through the very good performances exhibited by that work.
3. Analogically to [36], extracts meta-data from each HMM for generating a descriptor vector associated with each metric space. The good performances obtained in [36] demonstrate the accuracy and the efficiency of these descriptor vectors.

As a consequence, the proposed technique implicitly embeds the accuracy and the efficiency inherited from [8, 9, 36]. For these reasons, the experiments realized in the current paper do not aim at demonstrating the accuracy or the efficiency of the proposed technique. They are only intended to show how to use it in practice. The experiments realized in Section 5 have been intentionally limited to metric spaces containing geolocations or stars in the celestial sphere. We have avoided performing additional experiments for other datatypes ($\mathbb{R}^n$, trees, etc.) in order to reduce the paper length.

## 5 Experimental results

### 5.1 Experimental parameters

All the experiments realized in this section were executed on a personal computer with the following properties: (1) RAM: 16 GB, (2) Processors: Intel(R) Core(TM) i7-8665U CPU @1.9GHz 2.11GHz. Given that the proposed approach establishes a graduation in the neighborhood $\aleph_x$ of each instance $x$ of a dataset $X$, this neighborhood must on the one hand cover a surface that is large enough to semantically correspond to each level (slice) of the graduation. On the other hand, this surface shall not be too large because it just represents the neighborhood of an instance. For these reasons, a suitable value of the radius $R$ of $\aleph_x$ will be selected for each experience to fulfill these constraints. The values $p = 5$ and $p = 10$ have been experimented as the number of slices used for the division of $\aleph_x$. Given that the proposed approach relies on the technique proposed in [8] for generating HMMs, the following HMM-related settings adopted in [8] have been preserved here:

1. In [8], the number of states of each HMM was $(N + 1)$ and the authors experimentally selected the value $N = 50$. Consequently, the HMMs designed in the current work have 51 states.
2. In [8], the number of symbols of each HMM was also $(N + 1)$. Therefore, the number of symbols of the HMMs designed in the current work is also 51.

The maximum number of iterations of the *Baum-Welch* algorithm is limited to $\gamma = 50$ in the current paper considering the very large size of some experimental metric spaces. In the same way, the value $r = 100$ was selected for computing the stationary distribution of each HMM. The *Euclidean* and the *Manhattan* distances have both been selected as the distance $d$ between two vectors required for computing the $\sigma^d$ distance between two metric spaces using Eq. 15.

The experimental *GPS* coordinates used in this section were made publicly available in the year 2014 through the following official website of the French government: https://www.data.gouv.fr/. All the raw data used during the experiments of the current work, their corresponding sets of characteristic histograms and their complete URLs have been gathered in an online available archive file[20] No descriptor vector is presented in this section due to space constraints. Indeed, each descriptor vector has at least 51 components. Nevertheless, the descriptor vectors associated with all the metric spaces experimented in this section are also available online in the aforementioned archive file.

### 5.2 Examples of comparison of two datasets

In this section, we show how to use the proposed approach for comparing four French prefectures according to the *GPS* coordinates of the *communes* they contain and four stars constellations regarding the celestial coordinates of the *stars* they contain. There is no scientific justification for the choice of the selected prefectures and the selected stars constellations. They have been arbitrarily selected.

[20] http://perso-etis.ensea.fr/sylvain.iloga/index.html.

**Table 3** *σ-Euclidean* distance between the 4 selected prefectures in France in 2014 when $R = 10$ Km

| (a)- When $p = 5$ | | | | |
|---|---|---|---|---|
| | Cergy | Evry | Melun | Versailles |
| Cergy | 0 | 5.01 | 5.01 | 6.02 |
| Evry | 5.01 | 0 | 0.27 | 1.08 |
| Melun | 5.01 | 0.27 | 0 | 1.08 |
| Versailles | 6.02 | 1.08 | 1.08 | 0 |
| (b)- When $p = 10$ | | | | |
| Cergy | 0 | 0.61 | 1.24 | 1.24 |
| Evry | 0.61 | 0 | 1.11 | 1.11 |
| Melun | 1.24 | 1.11 | 0 | 0.11 |
| Versailles | 1.24 | 1.11 | 0.11 | 0 |

**Table 4** *σ-Manhattan* distance between the 4 selected prefectures in France in 2014 when $R = 10$ Km

| (a)- When $p = 5$ | | | | |
|---|---|---|---|---|
| | Cergy | Evry | Melun | Versailles |
| Cergy | 0 | 6.20 | 6.28 | 7.80 |
| Evry | 6.20 | 0 | 0.90 | 2.78 |
| Melun | 6.28 | 0.90 | 0 | 2.67 |
| Versailles | 7.80 | 2.78 | 2.67 | 0 |
| (b)- When $p = 10$ | | | | |
| Cergy | 0 | 1.57 | 2.99 | 2.99 |
| Evry | 1.57 | 0 | 2.79 | 2.81 |
| Melun | 2.99 | 2.79 | 0 | 0.34 |
| Versailles | 2.99 | 2.81 | 0.34 | 0 |

**Table 5** Durations (in seconds) required for computing the descriptor vectors of the 4 selected prefectures

| | Cergy | Evry | Melun | Versa. |
|---|---|---|---|---|
| $p = 5$ | 11.7 | 11.2 | 26.6 | 14.9 |
| $p = 10$ | 33.4 | 20.0 | 54.5 | 28.2 |

#### 5.2.1 Comparison of French prefectures

We decided to compare French prefectures which are considered here as datasets composed of '*communes*', each commune being described by its *GPS* coordinates. We have first downloaded the *GPS* coordinates of all the communes located in the four following prefectures in France: *Cergy*, *Evry*, *Melun* and *Versailles*. Table 2 presents the number of communes in each prefecture. During this experience, we fixed $R = 10$ Km. Therefore, each slice had a width of 2 Km when $p = 5$ and 1 Km when $p = 10$. The *σ-Euclidean* and *σ-Mahattan* distance between each pair of prefectures was computed and the results are respectively presented in Tables 3 and 4. According to these two tables, '*Melun*' and '*Evry*' are the nearest prefectures when $p = 5$, but when $p = 10$ the nearest prefectures are '*Melun*' and '*Versailles*'. This experimentally demonstrates the influence of the graduation of the neighborhood of each instance (i.e: the value of $p$) on the final comparison result. The durations (in seconds) required for computing the descriptor vectors of the 4 selected prefectures are presented in Table 5.

#### 5.2.2 Comparison of stars constellations

A constellation is a group of stars forming a recognizable pattern in the sky. A constellation is traditionally named according to its apparent form or identified with a mythological figure. This why there are constellations named *Dragon* and *Swan* because when their brightest stars are conveniently connected by imaginary lines in the sky, the resulting figures respectively look like each of these animals. Humans have thus divided the sky into 88 disjointed constellations in such a way that every star can belong to a unique constellation. A complete list of these constellations is provided in [37].

**Table 6** Number of stars in the 4 selected constellations recorded for the epoch 2000.0

| | Andro. | Swan | Dragon | Virgin |
|---|---|---|---|---|
| Stars | 1964 | 2736 | 2585 | 2723 |

A constellation is considered in the current experiment as a dataset composed of '*stars*' scattered in the sky, each star being described by its right ascension, its declination and its distance to the Earth.

Several catalogs listing the stars of the celestial sphere have yet been made publicly available. The most popular are the '*Hipparcos*', the '*Yale Bright Star*' and the '*Gliese*' catalogs. In most of these catalogs, each star is described by many properties including: its name, its right ascension, its declination, its distance to the Earth, its spectrum, its color index, etc. We have downloaded from [38] the '*HYG 3.0 database*' which contains all stars in *Hipparcos*, *Yale Bright Star*, and *Gliese* catalogs. In this database, the star's right ascensions and declinations are recorded for the epoch (year) 2000.0. Unfortunately, none of these catalogs gives the name of the constellation containing each star. To overcome this problem, we downloaded from [39] a C program which determines the name of the constellation containing any star given the celestial coordinates of that star and the epoch when these celestial coordinates were recorded.

We arbitrarily selected the four following constellations to perform comparisons: *Andromeda*, *Swan*, *Dragon* and *Virgin*. We remarked that in each selected constellation, there were some stars whose distance to the Earth was missing or dubious. These stars have been removed from the experimental database before the comparisons were performed. Table 6 presents the resulting number of stars in the selected constellations.

For this experience, we fixed $R = 500$ light-years. Therefore, each slice has a width of 100 light-years when $p = 5$ and 50 light-years when $p = 10$. The $\sigma$-*Euclidean* and $\sigma$-*Mahattan* distance between each pair of constellations was computed and the results are respectively

**Table 7** $\sigma$-*Euclidean* distance between the 4 selected constellations when $R = 500$ light-years

| (a)- When $p = 5$ | | | | |
|---|---|---|---|---|
| | Andro. | Swan | Dragon | Virgin |
| Andro. | 0 | 7.01 | 114 | 225 |
| Swan | 7.01 | 0 | 107 | 218 |
| Dragon | 114 | 107 | 0 | 111 |
| Virgin | 225 | 218 | 111 | 0 |
| (b)- When $p = 10$ | | | | |
| Andro. | 0 | 14 | 60 | 127 |
| Swan | 14 | 0 | 46 | 113 |
| Dragon | 60 | 46 | 0 | 67 |
| Virgin | 127 | 113 | 67 | 0 |

**Table 8** $\sigma$-*Manhattan* distance between the 4 selected constellations when $R = 500$ light-years

| (a)- When $p = 5$ | | | | |
|---|---|---|---|---|
| | Andro. | Swan | Dragon | Virgin |
| Andro. | 100 | 60.46 | 43.28 | 36.46 |
| Swan | 60.46 | 100 | 43.55 | 36.69 |
| Dragon | 43.28 | 43.55 | 100 | 52.34 |
| Virgin | 36.46 | 36.69 | 52.34 | 100 |
| (b)- When $p = 10$ | | | | |
| Andro. | 0 | 14.27 | 60.66 | 127.27 |
| Swan | 14.27 | 0 | 46.73 | 113.35 |
| Dragon | 60.66 | 46.73 | 0 | 67.49 |
| Virgin | 127.27 | 113.35 | 67.49 | 0 |

**Table 9** Durations (in seconds) required for computing the descriptor vectors of the 4 selected constellations

| | Andro. | Swan | Dragon | Virgin |
|---|---|---|---|---|
| $p = 5$ | 115.9 | 225.4 | 275.2 | 282.1 |
| $p = 10$ | 405.3 | 586.3 | 551.8 | 585.6 |

**Table 10** Comparison between '*Essonne*' (91) and '*Val d'Oise*' (95) using *GPS* data of 2014. Each department is considered as a dataset of '*venues*'

| | *σ-Euclidean* | *σ-Manhattan* |
|---|---|---|
| $p = 5$ | 15 | 16.45 |
| $p = 10$ | 2.01 | 3.11 |

**Table 11** Durations (in seconds) required for computing the descriptor vectors of '*Essonne*' (91) and '*Val d'Oise*' (95) when they are considered as datasets

| | Essonne | Val d'Oise |
|---|---|---|
| $p = 5$ | 97.5 | 106.4 |
| $p = 10$ | 83.0 | 70.0 |

**Table 12** Sizes of the subsets of departments *Essonne* and *Val d'Oise* in France in 2014 when each department is considered as a data collection composed of 4 subsets

| | *Essonne* | *Val d'Oise* |
|---|---|---|
| Football | 519 | 467 |
| Libraries | 99 | 95 |
| Pharmacies | 377 | 368 |
| Spectacle | 119 | 110 |

presented in Tables 7 and 8. These tables reveal that when $p = 5$, the constellations in the pairs (*Andromeda*,*Swan*) and (*Andromeda*,*Virgin*) are both eligible as the nearest. But this situation changes when $p = 10$ where only (*Andromeda*,*Swan*) exhibits the smallest distance for both, the *σ-Euclidean* and the *σ-Manhattan* distances. Consequently, the value $p = 10$ is experimentally more accurate for comparing these constellations. The durations (in seconds) required for computing the descriptor vectors of the 4 selected star constellations are available in Table 9.

## 5.3 Example of comparison of two data collections

### 5.3.1 French departments compared as datasets

In the current experience, we compare two French departments which are initially considered as datasets composed of '*venues*', each venue being characterized by its *GPS* coordinates. The two following French departments have been selected: '*Essonne*' and '*Val d'Oise*' respectively identified in France by the numbers 91 and 95. We have downloaded a file containing the *GPS* coordinates of 1114 venues for *Essonne* and 1040 venues for *Val d'Oise*. Equation 15 permitted us to obtain the results presented in Table 10. Table 11 contains the durations (in seconds) required for computing the descriptor vectors of these two departments in this context.

**Table 13** Comparison between '*Essonne*' (91) and '*Val d'Oise*' (95) using *GPS* data of 2014. Each department is considered as a data collection composed of 4 subsets

| | *σ-Euclidean* | *σ-Manhattan* |
|---|---|---|
| $p = 5$ | 8.90 | 21.08 |
| $p = 10$ | 16.10 | 28.23 |

**Table 14** Durations (in seconds) required for computing the descriptor vectors of '*Essonne*' (91) and '*Val d'Oise*' (95) when they are considered as data collections

| | Essonne | Val d'Oise |
|---|---|---|
| $p = 5$ | 469.0 | 449.7 |
| $p = 10$ | 629.1 | 435.7 |

### 5.3.2 French departments compared as data collections

We have now decided to perform a finer comparison of the two former French departments. Indeed, the venues of these departments are themselves football fields, municipal libraries, pharmacies and spectacle venues. Hence, each department is now considered as a data collection composed of elements described by their *GPS* coordinates and which belong to the following subtypes: '*football*', '*library*', '*pharmacy*' and '*spectacle*'. The sizes of the downloaded subsets composing these two departments are presented in Table 12 and Eq. 15 permitted us to obtain the results presented in Table 13. Table 14 shows the durations (in seconds) required for computing the descriptor vectors of these two departments when they are considered as data collections. The distances in Table 13 are more reliable than those in Table 10 because when selected departments are considered as data collections, a finer analysis of their contents is realized because the four subtypes of '*venues*' are taken into account during the analysis.

## 5.4 Theoretical time cost

The theoretical time cost required by the proposed approach for computing the descriptor vector $\vec{X}$ associated with a metric space $X$ can be derived from each of its main steps presented in Fig. 1 as follows:

1. The construction of the set $H_X$ of characteristic histograms consists in the gradual counting of the instances found in the neighborhoods of all the elements of $X$. Consequently, this step roughly takes the same time required for the non-gradual counting of the contents of these same neighborhoods realized in [9]. According to [9], this step runs in $(\alpha.|X|^2 + |X|.n)$ [21], where $n$ is the number of subtypes in $X$ and $\alpha$ is the number of arithmetic operations needed for the computation of the distance *dist* between two instances of $X$. We experimentally observed that this operation took at most 3 seconds for each experimental metric space.
2. Let $N$ be the number of states of the model. The HMM training requires $\gamma.(\sum_{k=1}^{|X|} p).N^2 = \gamma.|X|.p.N^2$ according to Section 3 because $|X|$ characteristic histograms, each having $p$ bins, are taken as inputs for this training. This was experimentally the most time consuming step.
3. The most time consuming operation realized during the meta-data extraction is the computation of the stationary distribution of the HMM which runs in $(r.N^3)$ according to Section 3. This step experimentally needed a couple of seconds or few minutes depending on the considered metric space.

Consequently, the theoretical time cost $\rho(\vec{X})$ required by the proposed approach for computing the descriptor vector $\vec{X}$ associated with a metric space $X$ is approximated by Eq. 16.

$$\rho(\vec{X}) = \theta\left[\alpha.|X|^2 + |X|.n + \gamma.|X|.p.N^2 + r.N^3\right] \quad (16)$$

## 6 Discussion

In this section, we discuss about some relevant related research problems that can be explored in future works.

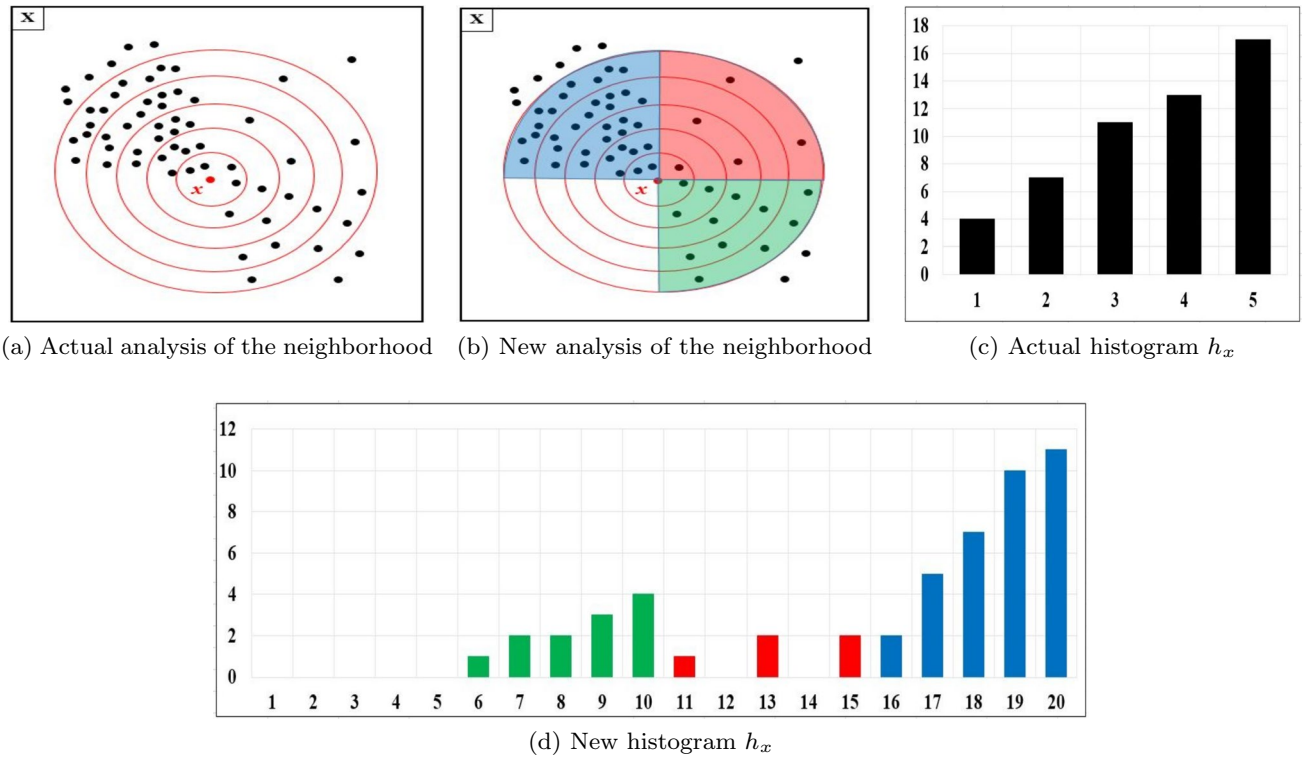
1. The proposed approach is highly customizable according to the four following generic parameters: the radius $R$ the neighborhood, the number $p$ of slices used for dividing this neighborhood, the maximum number of iterations $\gamma$ of the Baum-Welch algorithm and the distance $d$ between two vectors required in Eq. 15. Future work can analyze the impact of the variation of these generic parameters on the final result. The possibility of using two different radii, one for each metric space, during the analysis of the neighborhood of the instances can also be analyzed.
2. As it can be observed in the proposed methodology (Fig. 1), a descriptor vector $\vec{X}$ is initially generated for each dataset or data collection $X$ before performing the comparison. Consequently, the proposed method can be efficiently used in artificial intelligence tasks such as *classification* [40] and *clustering* [41]. Indeed, these two tasks both require descriptor vectors as inputs. One can for example suppose that all the French prefectures which are considered in Sect. 5.2.1 as datasets composed of *communes* are organized into the five following classes according to their rate of criminality: *extreme*, *high*, *normal*, *low*, *tiny*. This rate of criminality can first be attached to the descriptor vector associated with each prefecture, before to perform classification or clustering in a dedicated soft as WEKA [42].
3. *Multidimensional Scaling* (MDS) is a mathematical method which allows easier analysis of data by performing some sort of dimensionality reduction to map high-dimensional data into a lower-dimensional space in such a way that the distances between low-dimensional objects resemble the original similarity information in the high-dimensional space [43] [22]. MDS is not particularly required for the data actually experimented in the current paper because each geolocation on the Earth is characterized by only 2 features (the latitude and the longitude) and each star in the celestial sphere is only characterized by 3 features (the right ascension, the declination and the distance to the Earth). But if the input data are high-dimensional objects, we recommend the use of MDS to the 2D/3D space for example for visualizing the elements of each dataset as points scattered in the 2D/3D space.

## 7 Conclusions

The goal of this paper was to provide a solution for one of the major limitations of existing techniques for comparing two metric spaces which do not actually take into account the the way how elements are scattered inside the metric space. To achieve this goal, a gradual analysis of the neighborhood of each instance for each metric space is performed. This analysis is a refinement of the neighborhood's analysis performed in [9] where the dispersion of the instances inside the neighborhood was not taken into account. More formally, given a metric space $X$ and two user-defined numbers $R$ and $p$, the neighborhood $\aleph_x$ of each instance $x \in X$ is divided into $p$ slices of identical width $\frac{R}{p}$. The number of elements found in each slice are then used for generating the characteristic histogram $h_x$ associated with $x$. When this operation is repeated for all the instances in a metric space $X$, its corresponding set $H_X$ of characteristic histograms is obtained. The content of $H_X$

[21] See Section 4.3.4 of [9].

[22] See Sect. 3 of [43].




(a) Actual analysis of the neighborhood (b) New analysis of the neighborhood (c) Actual histogram $h_x$

(d) New histogram $h_x$

**Fig. 6** The new proposed approach for analyzing the neighborhood of an instance $x \in X$. Here, the neighborhood of $x$ initially divided into $p = 5$ slices, is additionally partitioned into $q = 4$ sectors, each sector being an arc of $\left(\frac{360°}{q}\right) = 90°$

is then taken as input for training the HMM $\lambda_X$ which learns the shapes and the bin variations of the histograms in $H_X$ using the technique proposed in [8]. Analogically to [36], meta-data related to the overall behavior of $\lambda_X$ regarding each symbol are then saved as the components of the descriptor vector $\vec{X}$ associated with $X$. Given two metric spaces $X$ and $Y$, the comparison between these metric spaces is performed through the comparison of their associated descriptor vectors $\vec{X}$ and $\vec{Y}$ using any existing distance/similarity between two vectors.

Experiments conducted on metric spaces containing geolocations and stars in the celestial sphere showed how to use the proposed approach in practice. The proposed approach inherits the accuracy and the efficiency of [8, 9, 36] on which it relies. But, an important drawback of the proposed technique is that its efficiency and its accuracy can be severely affected when it is applied on metric spaces whose population is very low. Indeed, when the population of a metric space is very low, the characteristic histograms derived from the neighborhood's analysis of its instances will be exclusively composed of very low bin values including a high number of zeros. As a consequence, the resulting HMM training will not be consistent because the HMM is trained for learning the visual shapes of the characteristic histograms which are not meaningful in this context.

The limitations of the proposed approach listed below can be handled in future work as follows:

1. In this paper, one characteristic histogram is constructed for each element of a metric space. This element-by-element construction of histograms may lead to *overfitting* during the model training. Indeed, the histograms associated with noises and outliers are also individually learned by the models. Additionally, in actual conditions, the training time of the HMMs increases exponentially when the size of the metric space is elevated. A possible issue for attenuating this limitation is to initially realize a clustering of the metric spaces using for example the *K-means* algorithm [4], before to associate one histogram to each of the $K$ centroids generated by the *K-means*. Future works should analyze the performances of this solution.
2. Consider two data collections $A$ and $B$. In this paper, we granted the same importance to every subtype during the computation of $\sigma^d(A, B)$. This is a limitation because it is basically possible for a user to grant different degrees of importance to these subtypes. As an example, a patient is very interested by the geolocations of hospitals and pharmacies, unlike a tourist who is more interested by the geolocations of hotels and airports. Thus, the value

of $\sigma^d(A,B)$ can depend on the degree of importance granted by the user to the various subtypes. It would therefore be very interesting in future work to propose a weighted version of the computation of $\sigma^d(A,B)$ which takes into account the importance (weight) granted by the user to each subtype.

3. Consider a metric space $X$ and an instance $x \in X$ whose neighborhood is divided into $p$ slices of identical width. In the current paper, all the neighbors of $x$ located in the $j^{th}$ slice of its neighborhood are counted for deriving the value of the $j^{th}$ bin of its characteristic histogram $h_x$. This is a limitation because this operation does not take into account the dispersion of the neighbors of $x$ inside the $j^{th}$ slice. However, if the dispersion of the neighbors of $x$ inside every slice are considered during the construction of $h_x$, the resulting HMM more accurately learns $X$.

   As an example, consider the metric space $X$ presented in Fig. 6a and consider the instance $x$ identified in that figure. When the principle proposed in the current paper is applied, the histogram $h_x = [4,7,11,13,17]$ presented in Fig. 6c is obtained. In order to take into account the dispersion of the neighbors of $x$ in every slice, we have partitioned the neighborhood of $x$ into $q = 4$ angular sectors, each sector being an arc of $\left(\frac{360°}{q}\right) = 90°$ as presented in Fig. 6b. This figure reveals the following previously unknown information about the neighborhood of $x$:

(a) The arc $[0° - 90°]$ is '*rarely*' populated.
(b) The arc $[90° - 180°]$ is '*densely*' populated.
(c) The arc $[180° - 270°]$ is '*not*' populated.
(d) The arc $[270° - 360°]$ is '*sparsely*' populated.

This limitation can be handled in future works by additionally partitioning into $q$ regular sectors the neighborhood of $x$ which was initially divided into $p$ slices, then to construct a partial characteristic histogram for each sector of each slice. The final characteristic histogram $h_x$ will be the concatenation of these $q$ partial characteristic histograms. For the instance $x$ identified in Fig. 6b, when we browse the sectors in the counterclockwise starting from sector $[180° - 270°]$, the following 20-bins characteristic histogram presented in Fig. 6d is obtained:$h_x = [0,0,0,0,0,1,2,2,3,4,1,0,2,0,2,2,5,7,10,11]$.

## References


1. Chen Shihyen, Ma Bin, Zhang Kaizhong (2009) On the similarity metric and the distance metric. Theoret Comput Sci 410(24–25):2365–2376
2. Konstantinos Kanakoglou (2012) The notion of abstract manifold: a pedagogical approach. arXiv preprint arXiv:1204.2191
3. Clementini Eliseo (2019) A conceptual framework for modelling spatial relations. Inform Technol Control 48(1):5–17
4. Hahsler Michael (2020) Cluster analysis: Basic concepts and algorithms. https://michael.hahsler.net/SMU/EMIS7331/slides/chap8_basic_cluster_analysis.pdf
5. shalizi Cosma (2009) Distance between clustering, hierarchical clustering. http://www.stat.cmu.edu/~cshalizi/350/lectures/08/lecture-08.pdf
6. Tatti N (2007) Distances between data sets based on summary statistics. J Machine Learning Res 8:131–154
7. Facundo Mémoli (2017) Distances between datasets. In Modern Approaches to Discrete Curvature, pages 115–132. Springer
8. Iloga, Sylvain and Romain, Olivier and Tchuenté Maurice (2018) An accurate hmm-based similarity measure between finite sets of histograms. Pattern Anal Appl 22(3):1079–1104
9. Iloga, Sylvain and Romain, Olivier and Tchuenté Maurice (2020) A sequential pattern mining approach to design taxonomies for hierarchical music genre recognition. Pattern Anal Appl 21(2):363–380
10. Iloga Sylvain (2021) Customizable HMM-based measures to accurately compare tree sets. Pattern Anal Appl 24(3):1149–1171
11. Cha Sung-Hyuk (2007) Comprehensive survey on distance/similarity measures between probability density functions. City 1(2):1
12. Mohamed Ahmed Zaid (2015) Correlation and regression analysis textbook. The Statistical, Economic and Social Research and Training Centre for Islamic Countries (SESRIC), Diplomatic Site, 6450
13. Swain Michael J, Ballard Dana H (1991) Color indexing. Int J Comput Vision 7(1):11–32
14. Deselaers Thomas, Keysers Daniel, Ney Hermann (2008) Features for image retrieval: an experimental comparison. Inf Retrieval 11(2):77–107
15. Haibin Ling, Kazunori Okada (2006) Diffusion distance for histogram comparison. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) 1:246–253
16. Rubner Yossi, Tomasi Carlo, Guibas Leonidas J (2000) The earth mover's distance as a metric for image retrieval. Int J Comput Vision 40(2):99–121
17. Kedem Dor, Tyree Stephen, Sha Fei, Lanckriet Gert, Weinberger Kilian Q (2012) Non-linear metric learning. Adv Neural Inf Process Syst 25:2573–2581
18. Bondy John Adrian, Murty Uppaluri Siva Ramachandra et al (1976) Graph theory with applications. Macmillan, London
19. Selkow Stanley M (1977) The tree-to-tree editing problem. Inf process lett 6(6):184–186
20. Pawlik Mateusz, Augsten Nikolaus (2015) Efficient computation of the tree edit distance. ACM Trans Database Syst (TODS) 40(1):1–40
21. Pawlik Mateusz, Augsten Nikolaus (2016) Tree edit distance: robust and memory-efficient. Inf Syst 56:157–173
22. Jiang Tao, Wang Lusheng, Zhang Kaizhong (1995) Alignment of trees-an alternative to tree edit. Theoret Comput Sci 143(1):137–148
23. Jesper Jansson, Andrzej Lingas (2001) A fast algorithm for optimal alignment between similar ordered trees. In Annual Symposium on Combinatorial Pattern Matching, Springer, pp. 232–240

24. Pekka Kilpel inen et al. (1992) Tree matching problems with applications to structured text databases. Helsingin yliopisto
25. Pekka Kilpel inen and Heikki Mannila (1995) Ordered and unordered tree inclusion. SIAM J Comput 24(2):340–356
26. Hoffmann Christoph M, O'Donnell Michael J (1982) Pattern matching in trees. J ACM 29(1):68–95
27. Zhang KZ, Shasha Dennis, Wang JTL (1994) Approximate tree matching in the presence of variable length don' t cares. J Algorithms 16(1):33–66
28. Tyng-Luh Liu and Davi Geiger (1999) Approximate tree matching and shape similarity. In Proceedings of the Seventh IEEE International Conference on Computer Vision, 1:456–462
29. Amir Amihood, Keselman Dmitry (1997) Maximum agreement subtree in a set of evolutionary trees: metrics and efficient algorithms. SIAM J Comput 26(6):1656–1669
30. Akutsu Tatsuya, Halldórsson Magnús M (2000) On the approximation of largest common subtrees and largest common point sets. Theoret Comput Sci 233(1–2):33–50
31. Nishimura Naomi, Ragde Prabhakar, Thilikos Dimitrios M (2000) Finding smallest supertrees under minor containment. Int J Found Comput Sci 11(03):445–465
32. Calculation of distance from gps coordinates in diadem. https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000019XnNSAU &l=en-US
33. Harald Schroer (2015) Distance between two stars. https://zenodo.org/record/29851/files/Distance_between_the_stars.pdf
34. Rabiner Lawrence R (1989) A tutorial on hidden markov models and selected applications in speech recognition. Proc IEEE 77(2):257–286
35. Jizhou Kang (2021) Stat 243: Stochastic process. https://bookdown.org/jkang37/stochastic-process-lecture-notes/lecture09.html
36. Sylvain Iloga, Alexandre Bordat, Julien Le Kernec, and Olivier Romain (2021) Human activity recognition based on acceleration data from smartphones using hmms. IEEE Access 9:139336–139351
37. List of the constellations. https://in-the-sky.org/data/constellations_list.php
38. The hyg database. http://www.astronexus.com/hyg
39. Identification of a constellation from position : vi/42. http://cdsarc.u-strasbg.fr/viz-bin/cat?cat=vi%2f42 &target=readme &
40. Krzystof Jajuga, Andrzej Sokolowski, Hans-Hermann Bock (2012) Classification, clustering, and data analysis: recent advances and applications
41. Glenn W Milligan, Stephen C Hirtle (2013) Clustering and classification methods
42. Witten Ian H, Frank Eibe (2005) Data mining: Practical machine learning tools and techniques. http://weka.sourceforge.net/
43. Damaševičius Robertas (2009) Analysis of components for generalization using multidimensional scaling. Fund Inf 91(3–4):507–522