Efficient interpretable variants of online SOM for large dissimilarity data

Jérôme Mariette[a], Madalina Olteanu[b], Nathalie Villa-Vialaneix[a]

[a]*MIAT, Université de Toulouse, INRA, 31326 Castanet-Tolosan, France*
[b]*SAMM, EA 4543, Université Paris 1, F-75634 Paris, France*

Abstract

Self-organizing maps (SOM) are a useful tool for exploring data. In its original version, the SOM algorithm was designed for numerical vectors. Since then, several extensions have been proposed to handle complex datasets described by (dis)similarities. Most of these extensions represent prototypes by a list of (dis)similarities with the entire dataset and suffer from several drawbacks: their complexity is increased - it becomes quadratic instead of linear -, the stability is reduced and the interpretability of the prototypes is lost.

In the present article, we propose and compare two extensions of the stochastic SOM for (dis)similarity data: the first one uses a dimension reduction in a feature space defined by the (dis)similarity, while the second one takes advantage of the online setting in order to maintain a sparse representation of the prototypes at each step of the algorithm. Our contributions to the analysis of (dis)similarity data with topographic maps are thus twofolds: first, we present a new version of the SOM algorithm which ensures a sparse representation of the prototypes through online updates. Second, this approach is compared on several benchmarks to a standard dimension reduction technique (K-PCA), which is itself adapted to large datasets with the Nyström approximation.

Results demonstrate that both approaches lead to reduce the prototypes dimensionality while providing accurate results in a reasonable computational time. Selecting one of these two strategies depends on the dataset size, the need to easily interpret the results and the computational facilities available. The conclusion tries to provide some recommendations to help the user making this choice.

Keywords:

SOM, Sparse methods, Kernel, dissimilarity, K-PCA, Nyström

1. Introduction

1.1. State-of-the art on SOM for (dis)similarity data

Over the years, the self-organizing map (SOM) algorithm [1] was proved to be a powerful and convenient tool for clustering and visualizing data [2, 3, 4, 5, 6]. While the original algorithm had been designed for numerical vectors, the available data in the applications became more and more complex, being frequently too rich to be described by a fixed set of numerical attributes only. This is the case, for example, when data are described by relations between objects (individuals involved in a social network) or by measures of resemblance/dissemblance which are context specific (see [7, 8] for similarities between categorical sequences, [9] for similarities between microbial diversity distributions, [10] for similarities in gene expression data).

During the past twenty years, the SOM algorithm was extended to handle non numerical data. For example, SOM was adapted to categorical data analysis, by using a method similar to Multiple Correspondence Analysis in [11]. Another solution, called median SOM [12], addressed the issue of data described by pairwise relations (similarities or dissimilarities): in this solution, the standard computation of the prototypes is replaced by an approximation within the original dataset. However, as prototypes are chosen among the data, their representation is very restrictive. In order to increase the flexibility of the prototypes, [13] proposed to represent a class by several prototypes, all chosen among the original dataset. But, this method seriously increases the computational time, while prototypes remain restricted to the original dataset and may generate possible sampling or sparsity issues.

A very different approach to handle relational data consists in relying on a (pseudo-)Euclidean framework, following the results of [14] (for data described by a kernel) or of [15] (for dissimilarity data). This approach was developed in the framework of kernel SOM (see [16] for the online version and [17] for the batch version), and in the framework of relational SOM (see [18] for the online version and [19] for the batch version). Kernel SOM and relational SOM are equivalent if the dissimilarity in relational SOM is the squared distance induced by the kernel. The key idea of this approach is to express prototypes as convex combinations of the images of the original data $(x_i)_{i=1,\dots,n}$ in a (pseudo-)Euclidean space in which the data are (implicitly) embedded by the kernel (or the dissimilarity): as stated in [20, 21],

this solution yields several drawbacks due to the large dimensionality of the embedding space (which is equal to the number of observations, n). Firstly, the complexity (in n) is strongly increased and becomes at least quadratic. As stressed in [19], algorithms will be slow for datasets with 10,000 observations and impossible to run on a normal computer for 100,000 input data. Secondly, the results are highly unstable: especially in the online (also called stochastic) version of the algorithm, two different runs of the method can provide very different results. Thirdly, one of the most important features of the SOM algorithm is lost: in standard numerical SOM, clusters are represented by a single prototype valued in the data space. These prototypes help to interpret the obtained clusters and thus the overall map organization. In kernel/relational SOM, prototypes are given as n coefficients that correspond to a resemblance with each of the n observations: they do not correspond themselves to an observation in the original data space and as such, prototypes are not much more informative than the clustering itself.

In conclusion, kernel and relational extensions of the standard SOM algorithm are hardly practicable when the dataset is large. This is due partly to the number of observations, but also to the dimensionality of the (embedded) data which is directly related to this number. To address this issue, strategies usually used to handle large datasets or datasets with a high dimensionality are useful and they can even be combined.

1.2. Review of methods for large datasets and high dimensional datasets

Different strategies were developed and are available in the literature to handle large datasets (when the number of observations is large) and high-dimensional datasets (when the dimension of the dataset is large). For large datasets, standard approaches include i) *divide and conquer approaches* [22, 23, 24] in which data are split into several bits of data which are processed separately. The results are aggregated afterwards to obtain a final solution which is supposed to well approximate the solution that would have been obtained if the entire dataset had been processed at once; ii) *subsampling methods* [25, 26, 27, 28, 29], which consist in using a restricted subset (usually carefully designed) of the original data, in order to approximate the solution that could have been obtained with the entire dataset and iii) *online* updates [30, 31], in which the results are updated with sequential steps, each having a low computational cost.

A particular case of the subsampling strategy is the Nyström approximation [32], which consists in sampling a small number of rows/columns in

square matrices and in obtaining an approximation of its eigendecomposition at a very reduced computational cost. The eigendecomposition is even exact when the matrix is of low rank (when the size of the subsample is larger than the rank of the matrix). This method is frequently used for kernel and dissimilarity-based algorithms.

For high-dimensional data, the strategies are a bit different and include i) *sparse approaches* [33, 34], in which a subset of the variables is selected to build the final predictive model. This subset can be obtained from sequential exploration (stepwise strategies), from approximation heuristics or by using a sparse penalty term within the model (LASSO); ii) *dimension reduction (DR)* techniques, that can be linear (PCA for instance or random projections as in [35]) or nonlinear [36]. DR methods embed the data in a small dimensional space and are usually mainly used for visualization and exploratory analysis. However, if the embedding can be obtained at a low cost, it can be used as an approximation of the high-dimensional dataset on which more costly algorithms may be applied. SOM itself is a dimension reduction method but, as stressed before, the computational complexity of its kernel and relational versions is high. Finally, a particular case of DR techniques is *model-based clustering* methods, which use mixture distributions and embed the data in a low-dimensional subspace that is the best suited for clustering (see [37] for a review).

1.3. Kernel/relational extensions for large datasets

Several extensions for kernel and relational data of the standard SOM algorithm, or of related kernel/relational algorithms (such as, *e.g.*, k-means, LVQ, topographic maps...) have already been proposed in the literature. They use ideas coming from the strategies handling large and/or high-dimensional datasets cited above. Most of them seek a simplified/sparse representation of the prototypes and/or a reduced computational time.

In the relational k-means framework, [38] proposed a sparse extension of the batch algorithm: every prototype is represented by at most K (K fixed) observations by cluster, that are selected at each step of the algorithm. In the supervised framework, [21] used a similar strategy for batch LVQ, by selecting the most representative observations (with different methods to obtain them, including approximation heuristics and L^1 penalty) in every cluster and at each step of the algorithm. A similar method was used in [39], combined with the Nyström approximation of the LVQ algorithm, in order to obtain sparse prototypes at very low computational cost. The Nyström approximation

was also used for obtaining faster versions of topographic mapping methods [40, 41] and for reducing the computational cost of the clustering. Another subsampling strategy was used in a nonlinear (kernel) DR framework to allow processing large datasets, in [42].

However, these approaches do not lead to a simplified (and thus interpretable) representation of the prototypes. Furthermore, all of them are restricted to the batch framework and most of them are performed after each iteration of a batch algorithm, *i.e.*, after all observations have been processed at least once. An alternative to these methods consists in splitting the data into several subsets on which independent algorithms are trained: in [19], the complexity is reduced using iterative "patch clustering" that mixes "divide and conquer" and "online updates" methods. First, the data are split into B patches of size n_B ($\ll n$, B fixed). A prototype-based clustering algorithm in batch version (neural gas or SOM) is then run on a patch $\mathcal{P}_t$. The resulting prototypes, which may be viewed as compressed representations of the data already seen, are then added as new data points to the next patch, $\mathcal{P}_{t+1}$. Moreover, the full vector of coefficients is replaced by the Q closest input data (Q fixed). With this method, linear time and constant space representation are obtained but the sequential training may influence the final result since all observations are not processed evenly.

In the same line of thoughts, [43] propose a bagging approach for kernel SOM. Data are split into B subsamples of size n_B ($\ll n$, B fixed), the online kernel SOM is trained on each subsample and, after training, the most representative Q observations are chosen for each prototype (Q fixed). Eventually, a final map is trained on the resulting most representative observations. In this method, parallel computing techniques can be used for reducing the computational time. However, the results of the B trained SOMs are not used as such but only to select the most representative observations in the dataset.

1.4. Contributions of the article

In the present article, we propose and compare two methods to obtain sparse prototypes and a reduced computational cost in the online SOM algorithm. The first one uses a reduction dimension in the embedding space, which can be efficiently performed with Nyström technique. This method combines ideas coming from the high dimension and the large data problems. However, it is not specific to the online setting. We thus compared it with another approach which takes advantage of the online framework to

provide sparse prototypes at all iteration steps of the algorithm: the coefficients are interpreted similarly to an amount of mass and the most important observations are selected by using a fixed global probability mass, ν, such that only the largest coefficients summing to at most ν are kept. This second proposal also takes ideas from large data problems (online updates) and from high dimension (variable selection with sparsity).

For the sake of simplicity, most of the paper is written in the framework of kernel SOM but its extension to relational SOM is straightforward and briefly explained in Section 5. The rest of the paper is organized as follows: Section 2 recalls the online kernel SOM (online K-SOM). Section 3 describes the dimension reduction approach for online K-SOM while Section 4 presents the direct sparse version of online K-SOM. Comparisons and numerical experiments are reported in Section 6.

2. Kernel SOM (K-SOM)

This section describes the theoretical background of the online version of the SOM algorithm and its extension to data described by a kernel. In the following, we consider a set of observations $(x_i)_{i=1,\dots,n}$ which take values in an arbitrary space $\mathcal{X}$. $\mathcal{X}$ is equipped with a kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that provides pairwise similarity between the observations, $K_{ij} := K(x_i, x_j)$. K is assumed symmetric ($K_{ij} = K_{ji}$) and positive ($\forall\, N \in \mathbb{N},\ \forall\, (\alpha_i)_{i=1,\dots,N} \subset \mathbb{R},\ \forall\, (x_i)_{i=1,\dots,N} \subset \mathcal{X},\ \sum_{i,i'} \alpha_i \alpha_{i'} K_{ii'} \geq 0$). According to [14], K is the dot product in a uniquely defined Hilbert space $(\mathcal{H}, \langle ., . \rangle)$ of the images of x_i and x_j by a uniquely defined feature map $\phi : \mathcal{X} \to \mathcal{H}$:

$$K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle.$$

The SOM algorithm aims at mapping the input data onto a low dimensional grid (usually a two-dimensional rectangle), composed of U units, each of them described by a prototype p_u, $u = 1, \dots, U$. The units are related together by a neighborhood relationship H, expressed as a function of the distance between the units on the grid, $d(u, u')$, $H : \mathbb{R} \to \mathbb{R}$. Classically, H is chosen such that $H(0) = 1$, $\lim_{x \to +\infty} H(x) = 0$ and is decreased during the training. Also, the distance on the grid, d, may be chosen as the length of the shortest path between the units or as the Euclidean distance between the coordinates of the units positioned at $(1, 1)$, $(1, 2)$, $\dots$, (m, m') (where $m \times m' = U$).

In standard (numerical) SOM, the data take values in $\mathcal{X} = \mathbb{R}^d$ and the kernel is the standard dot product in this space $K_{ij} = x_i^\top x_j$. In this case, the prototypes are also valued in $\mathbb{R}^d$. In kernel SOM, $\mathcal{X}$ is arbitrary and prototypes are not easily defined in this space, which may not be Euclidean or equipped with standard operations such as the sum. To allow for a more flexible representation of the prototypes (*i.e.*, a representation which is not restricted to the data already observed, $(x_i)_i$), the implicit feature space $(\mathcal{H}, \langle ., . \rangle)$ is used and the algorithm is re-defined in this space. The prototypes are expressed as convex combinations of the images by ϕ of the original data: $p_u = \sum_{i=1}^n \beta_{ui}\phi(x_i)$, $\beta_{ui} \geq 0$ and $\sum_i \beta_{ui} = 1$. As said before, the feature map is uniquely defined from the kernel K. However, it is usually not explicitly given and thus, all calculations are based on the values of the coefficients $\boldsymbol{\beta}_u = (\beta_{u1}, \ldots, \beta_{un}) \in \mathbb{R}^n$ only.

The kernel version of online SOM [16, 38] is thus directly derived from the computations of the standard numerical SOM performed in the feature space $\mathcal{H}$: the norm and the dot product in $\mathcal{H}$ can be obtained using the kernel, without the need of explicitly knowing $\mathcal{H}$ or ϕ (this is the so-called "kernel trick"). Online K-SOM is provided in Algorithm 1. In this algorithm, $\mu(t)$ usually vanishes at the rate $\frac{1}{t}$ and the final clustering is defined as $(\mathcal{C}_u)_{u=1,\ldots,U}$, where $\mathcal{C}_u = \{x_i : \ f(x_i) = u\}$ with $f := f^T$.

The issue with kernel SOM is that, when n is large, the total number of coefficients β_{ui} to learn is equal to $n \times U$, which yields a complexity of $\mathcal{O}(n^2U)$ (for one iteration) instead of $\mathcal{O}(dU) + \mathcal{O}(ndU)$ for the standard numerical SOM in $\mathbb{R}^d$. Hence, this algorithm cannot be used to analyze datasets with more than a few thousands observations. Also, the representation of the prototypes is so flexible that the results are highly unstable with different final clusterings and different prototypes for each run of the algorithm. Finally, as stated in [20], one of the main challenges of this kind of algorithm is that the easiness in interpreting the resulting map is lost: in standard SOM, prototypes are easily interpreted because they are described by values of well known variables which are the variables also describing the observations in the input dataset. In kernel SOM, the prototypes are characterized by their proximity to all individuals in the dataset. Hence, interpreting the clustering requires to be able to understand the relationships of the prototypes with all individuals, which is probably as difficult as analyzing the entire dataset directly.

Algorithm 1 Online kernel SOM

For all $u = 1, \ldots, U$ and $i = 1, \ldots, n$, initialize $(\beta_{ui}^0)_{u,i}$ such that $\beta_{ui}^0 \geq 0$ and $\sum_i^n \beta_{ui}^0 = 1$.

for $t = 1, \ldots, T$ **do**

Randomly choose an input x_i

Assignment step: find the unit of the prototype closest to $\phi(x_i) \in \mathcal{H}$

$$f^t(x_i) \leftarrow \arg \min_{u=1,\ldots,U} \|p_u^{t-1} - \phi(x_i)\|^2$$

which is equivalent to minimize, over u, $\sum_{j,j'} \beta_{uj}^{t-1} \beta_{uj'}^{t-1} K_{jj'} - 2 \sum_j \beta_{uj}^{t-1} K_{ij}$;

Representation step: $\forall\, u = 1, \ldots, U$,

$$\boldsymbol{\beta}_u^t \leftarrow \boldsymbol{\beta}_u^{t-1} + \mu(t) H^t(d(f^t(x_i), u)) \left(\mathbf{1}_i - \boldsymbol{\beta}_u^{t-1}\right)$$

where $\mathbf{1}_i$ is a vector with a single non null coefficient at the ith position, equal to one.

end for

3. A dimension reduction technique for K-SOM

This section describes a first approach to reduce the dimensionality of the prototypes, thus improving the computational complexity as well as the interpretability of the prototypes. Our proposal is to rely on a preprocessing of the data based on PCA in the feature space (Kernel PCA, denoted by K-PCA and described in Section 3.1) and then to express the SOM algorithm in the subspace of the feature space $\mathcal{H}$ which is spanned by the first eigenvectors of the K-PCA (Section 3.2). Finally, the computational complexity of the approach can be further reduced by performing the K-PCA thanks to the Nyström approximation (Section 3.3).

3.1. Kernel PCA (K-PCA)

K-PCA, introduced in [44], is a PCA analysis performed in the feature space $(\mathcal{H}, \langle ., . \rangle)$ induced by the kernel. A centered data matrix is first com-

puted using:

$$\begin{aligned}\widetilde{K}_{ij} &:= \left\langle \phi(x_i) - \frac{1}{n}\sum_{l=1}^{n}\phi(x_l), \phi(x_j) - \frac{1}{n}\sum_{l=1}^{n}\phi(x_l) \right\rangle \\ &= K_{ij} - \frac{1}{n}\sum_{l=1}^{n}(K_{il}+K_{jl}) + \frac{1}{n^2}\sum_{l,l'=1}^{n}K_{ll'}, \end{aligned} \tag{1}$$

which yields to the modified centered kernel $\widetilde{K} = K - \frac{1}{n}\mathbf{I}_n^T K - \frac{1}{n}K\mathbf{I}_n + \frac{1}{n^2}\mathbf{I}_n^T K\mathbf{I}_n$, in which $\mathbf{I}_n$ is the vector with n entries equal to 1. The eigenvectors $(\boldsymbol{\alpha}_k)_{k=1,\dots,n} \in \mathbb{R}^n$ and the corresponding eigenvalues $(\lambda_k)_{k=1,\dots,n}$ are obtained from this centered matrix by solving the following eigenvalue problem:

$$\widetilde{K}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}. \tag{2}$$

This problem is equivalent to finding eigenvectors, in the feature space $\mathcal{H}$, of the covariance matrix of the (centered) images of the original data by ϕ (the feature map associated to the centered kernel $\widetilde{K}$). These vectors, $(a_k)_{k=1,\dots,n} \in \mathcal{H}$ lie in the span of $\{\phi(x_i)\}_{i=1,\dots,n}$ and can be expressed as:

$$a_k = \sum_{i=1}^{n}\alpha_{ki}\phi(x_i)$$

where $\boldsymbol{\alpha}_k = (\alpha_{ki})_{i=1,\dots,n}$. $\boldsymbol{a}_k = (a_{ki})_{i=1,\dots,n}$ are orthonormal in $\mathcal{H}$:

$$\forall\, k,k',\ \langle a_k, a_{k'}\rangle = \boldsymbol{\alpha}_k^\top \widetilde{K}\boldsymbol{\alpha}_{k'} = \delta_{kk'} \qquad \text{with} \qquad \delta_{kk'} = \begin{cases} 0 & \text{if } k \neq k' \\ 1 & \text{otherwise} \end{cases}.$$

The principal components are the coordinates of the projections of the images of the original data, $(\phi(x_i))_i$ onto the eigenvectors $(a_k)_{k\geq 1}$ which can be expressed as:

$$\langle a_k, \phi(x_i)\rangle = \sum_{j=1}^{n}\alpha_{kj}\widetilde{K}_{ji} = \widetilde{K}_{i.}\boldsymbol{\alpha}_k,$$

where $\widetilde{K}_{i.}$ is the i-th row of the kernel $\widetilde{K}$. According to Equation (2), we thus obtain that $\langle a_k, \phi(x_i)\rangle = \lambda_k\alpha_{ki}$. The original data are projected on these axes with:

$$\mathcal{P}_{a_k}(\phi(x_i)) = \langle a_k, \phi(x_i)\rangle a_k = (\lambda_k\alpha_{ki})a_k.$$

The result of K-PCA can be used to approximate the data in a reduced space by selecting p axes $(a_k)_{k=1,\dots,p}$ in the feature space $\mathcal{H}$ (with $p \ll n$), on which the data can be projected.

3.2. K-PCA SOM

We suppose in this section that the kernel K is centered. Otherwise, without loss of generality, the centering procedure described in Equation (1) can be applied.

The main idea developed in this section is to define the U prototypes of the SOM in $A = \text{Span}\{a_1, \ldots, a_p\}$ instead of the entire feature space. A prototype can thus be written as:

$$p_u = \sum_{k=1}^{p} \beta_{uk} a_k$$

with $\beta_{uk} \geq 0$ and $\sum_k \beta_{uk} = 1$ and the two steps of the SOM algorithm can thus be rewritten as:

- *assignment step*: when the observation x_i is picked at random, it is affected to:

$$f(x_i) := \arg\min_u \|p_u - \phi(x_i)\|^2$$

in which:

$$\begin{aligned}
\|p_u - \phi(x_i)\|^2 &= \sum_{k,k'=1}^{p} \beta_{uk}\beta_{uk'}\langle a_k, a_{k'}\rangle - 2\sum_{k=1}^{p} \beta_{uk}\langle a_k, \phi(x_i)\rangle + \|\phi(x_i)\|^2 \\
&= \sum_{k,k'=1}^{p} \beta_{uk}\beta_{uk'}\langle a_k, a_{k'}\rangle - 2\sum_{k=1}^{p} \beta_{uk}\lambda_k\alpha_{ki} + \|\phi(x_i)\|^2 \\
&= \boldsymbol{\beta}_u^\top \boldsymbol{\beta}_u - 2\boldsymbol{\beta}_u^\top \Lambda \boldsymbol{\alpha}_{.i} + \|\phi(x_i)\|^2, \qquad (3)
\end{aligned}$$

where $\boldsymbol{\beta}_u = (\beta_{uk})_{k=1,\ldots,p}$, $\Lambda = \text{Diag}\,(\lambda_1, \ldots, \lambda_p)$ and $\boldsymbol{\alpha}_{.i}$ is the i-th column of $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p]^\top$. This step is thus equivalent to minimize $\boldsymbol{\beta}_u^\top \boldsymbol{\beta}_u - 2\boldsymbol{\beta}_u^\top \Lambda \boldsymbol{\alpha}_{.i}$ over $u \in \{1, \ldots, U\}$ and is also equivalent to minimize $\|p_u - \mathcal{P}_A(\phi(x_i))\|^2$ over u. This result was expected since, by the definition of $\mathcal{P}_A(\phi(x_i))$, we have that $\|p_u - \phi(x_i)\|^2 = \|p_u - \mathcal{P}_A(\phi(x_i))\|^2 + \|\mathcal{P}_A(\phi(x_i)) - \phi(x_i)\|^2$, in which the second term does not depend on u;

- *representation step*: the gradient descent like step is given by

$$\begin{aligned}
p_u &= p_u + \mu H(d(f(x_i), u))\,(\mathcal{P}_A(\phi(x_i)) - p_u) \\
&= \sum_k \beta_{uk} a_k + \mu H(d(f(x_i), u)) \left(\sum_k (\lambda_k \alpha_{ki}) a_k - \sum_k \beta_{uk} a_k \right),
\end{aligned}$$

which is equivalent to update the coefficients as:

$$\boldsymbol{\beta}_u = \boldsymbol{\beta}_u + \mu H(d(f(x_i), u)) \left(\Lambda \boldsymbol{\alpha}_{.i} - \boldsymbol{\beta}_u\right).$$

This approach is simply the standard (numerical) SOM with entries the $n \times p$ matrix $\boldsymbol{\alpha}^\top \Lambda$. The computational complexity of this approach is thus reduced from $\mathcal{O}(n^2UT)$ (online K-SOM, T iterations) to $\mathcal{O}(pUT) + \mathcal{O}(npU)$ (online numeric SOM in $\mathbb{R}^p$, cost of the T iterations and cost of the final clustering computation), once the K-PCA is given. Hence, since T is generally of the order of n, this gives a linear complexity for the algorithm. However, K-PCA itself can have a large complexity. This issue is addressed in the next section.

The interpretation of the prototypes is done similarly as for a standard PCA: the axes a_1, ..., a_p of the prototypes are interpreted with respect to the observations $(x_i)_i$ which contribute most to their definition. Then, prototypes are interpreted by means of their coefficients on these axis.

3.3. K-PCA from Nyström approximation

Kernel-based methods, and especially K-PCA, do not scale well when the number of observations, n, is large. For instance, the computational complexity of the eigendecomposition of K is $\mathcal{O}(n^3)$. An effective approach for improving the scalability of kernel methods is the Nyström approximation [32] and a similar technique is used in [41] for improving the computational cost of topographic maps for dissimilarity data. In the latter reference, the authors use the Nyström approximation to reduce the computational cost of $\|p_u - \phi(x_i)\|^2$ from $\mathcal{O}(n^2)$ to $\mathcal{O}(m^2 n)$, in which m is a number of observations taken at random within the original dataset. m is supposed to be small compared to n and close to the kernel rank.

In the approach introduced in Section 3.2 and unlike article [41], it is the pre-processing step which is addressed by the Nyström approximation, while the computational cost of calculating the clustering of the map is handled by the use of the numerical version of the algorithm based on the K-PCA pre-processing. More precisely, the eigendecomposition of a kernel matrix K (which is supposed to be centered) is approximated by selecting m observations, $\mathcal{T}_m$ among $(x_i)_{i=1,\ldots,n}$, and by using the eigendecomposition of the reduced matrix $K^{(m)} = (K(x_i, x_j))_{i,j \in \mathcal{T}_m}$. In practice, the selected observations $\mathcal{T}_m$ are chosen at random, although more efficient sampling techniques such as the ones described and evaluated in [45] could also be used.

If the eigenvalues and the (orthonormal) eigenvectors of $K^{(m)}$ are denoted by $(\lambda_k^{(m)})_k$ and $(v_k^{(m)})_k$ respectively, then the eigenvalues and (orthonormal) eigenvectors of K are given by

$$\lambda_k \simeq \frac{n}{m}\lambda_k^{(m)} \qquad \text{and} \qquad v_{ki} \simeq \sqrt{\frac{m}{n}}\frac{1}{\lambda_k^{(m)}}K_{i.}^{(n,m)}v_k^{(m)},$$

with $K_{i.}^{(n,m)}$ the i-th row of the matrix $K^{(n,m)} = (K(x_j, x_{j'}))_{j=1,\ldots,n,\, j'\in\mathcal{T}_m}$. If the rank of $K^{(m)}$ is equal to the rank of the original matrix K, the approximation even becomes an equality. Then, assuming that the kernel K (which is supposed to be centered) is known or at least that any of the pairs $K(x_i, x_j)$ $(i,\, j = 1, \ldots, n)$ can be computed at low cost, the K-PCA requires to obtain the entries $(\lambda_k\alpha_{ki})_{k=1,\ldots,p} \in \mathbb{R}^p$ for all $i = 1, \ldots, n$, where $(\boldsymbol{\alpha}_k)_k$ are the eigenvectors of K which are orthogonal with respect to the norm induced by the kernel. We can easily show that

$$\boldsymbol{\alpha}_k = \frac{v_k}{\sqrt{\lambda_k}} \tag{4}$$

because, by definition, as v_k are eigenvectors of K, so are $\boldsymbol{\alpha}_k$ and, in addition, using the equality of Equation (4), we have that:

$$\boldsymbol{\alpha}_k^\top K\boldsymbol{\alpha}_{k'} = \frac{1}{\sqrt{\lambda_k}\sqrt{\lambda_{k'}}}v_k^T K v_{k'} = \frac{1}{\sqrt{\lambda_k}\sqrt{\lambda_{k'}}}v_k^T\lambda_{k'}v_{k'} = \delta_{kk'}.$$

Therefore, K-PCA can be computed with entries the rows of the $n \times p$ matrix $\boldsymbol{\alpha}^\top\Lambda$ with $\forall\, i = 1, \ldots, n$ and $\forall\, k = 1, \ldots, p$,

$$\lambda_k\alpha_{ki} = \sqrt{\lambda_k}v_{ki} = \frac{1}{\sqrt{\lambda_k^{(m)}}}K_{i.}^{(n,m)}v_k^{(m)} = K_{i.}^{(n,m)}\boldsymbol{\alpha}_k^{(m)} \tag{5}$$

with $\boldsymbol{\alpha}_k^{(m)} = \frac{v_k^{(m)}}{\sqrt{\lambda_k^{(m)}}}$. This simplified K-PCA is a good approximation of the K-PCA with kernel K. The complexity of the preprocessing is reduced from $\mathcal{O}(n^3)$ (standard K-PCA) to $\mathcal{O}(m^3)+\mathcal{O}(nm^2)$ (respectively, K-PCA based on the reduced kernel and approximation). The complete algorithm is provided in Algorithm 2.

Algorithm 2 Online K-PCA SOM

Nyström approximation of K-PCA

Select at random m observations $\mathcal{T}_m = \{x_{i(1)}, \dots, x_{i(m)}\}$ from the original dataset

Compute the first p eigenvalues and orthonormal eigenvectors of $K^{(m)}$, $(\lambda_k^{(m)})_{k=1,\dots,p}$, $(v_k^{(m)})_{k=1,\dots,p}$ and obtain, for $k = 1, \dots, p$, $\boldsymbol{\alpha}_k^{(m)} = \frac{v_k}{\sqrt{\lambda_k}}$

Compute $\left(K^{(n,m)}\boldsymbol{\alpha}_k^{(m)}\right)_{k=1,\dots,p}$, which is an $n \times p$ matrix $B = (b_{ik})_{i=1,\dots,n,\, k=1,\dots,p}$

K-PCA SOM

Initialize prototypes randomly: $\forall\, u = 1, \dots, U$, $\boldsymbol{\beta}_u^0 \in [0,1]^p$ and $\sum_{k=1}^p \beta_{uk}^0 = 1$

for $t = 1, \dots, T$ **do**

  Randomly choose an input $i \in \{1, \dots, n\}$

  Assignment step: find the unit of the prototype closest to i-th row of B, $\mathbf{b}_i$:

$$f^t(x_i) \leftarrow \arg\min_{u=1,\dots,U} \|\boldsymbol{\beta}_u^{t-1} - \mathbf{b}_i\|^2_{\mathbb{R}^p}$$

  Representation step: $\forall\, u = 1, \dots, U$,

$$\boldsymbol{\beta}_u^t \leftarrow \boldsymbol{\beta}_u^{t-1} + \mu(t) H^t(d(f^t(x_i), u)) \left(\mathbf{b}_i - \boldsymbol{\beta}_u^{t-1}\right)$$

end for

4. Direct sparse K-SOM

In this section, we present an alternative approach to obtain sparse prototypes while taking advantage of the online updates of the stochastic version of the K-SOM algorithm. Here, unlike in K-PCA SOM, the prototypes are directly written as convex combinations of the observations, but, in this case, they are restricted to the input data already fed to the algorithm and, more particularly, to the most important of them. This algorithm has already been described in the framework of dissimilarity data and tested on restricted datasets in [46].

4.1. Description of the approach

To ensure sparsity through the algorithm, the prototypes are initialized at random among the input data. Thus, the method first selects at random U observations that are used as initial values for the U prototypes. Then, at a given step t of the algorithm, a prototype is written as a convex combination of the most important past observations: if $I_u(t-1)$ denotes the set of the most important observations selected for prototype p_u before step t, p_u writes $p_u = \sum_{j \in I_u(t)} \beta_{uj} \phi(x_i)$. Finally, the distance in the feature space $\mathcal{H}$ between a new input, x_i selected at random, and p_u is given by:

$$\|\phi(x_i) - p_u\|^2 = \sum_{j,j' \in I_u(t-1)} \beta_{uj}\beta_{uj'} K(x_j, x_{j'}) - 2 \sum_{j \in I_u(t-1)} \beta_{uj} K(x_j, x_i) + K(x_i, x_i). \tag{6}$$

In order to maintain prototypes as sparse combinations of the input data, they are periodically updated and the most important coefficients only are kept. The update instants may be performed throughout the iterations using various strategies: for instance, they can be uniformly distributed during the learning process or distributed according to some geometric distribution. The parameter of the geometric distribution may be fixed, during the whole training, or varying (in ascending or descending fashion) with the iterations. We ran several simulations on various data sets, using these four scenarios (results not shown). The results were globally similar, with a slight advantage for the "ascending random" strategy. In this case, the parameter of the geometric distribution was $\gamma_t = \frac{1-\mu(t)}{\kappa}$, where κ is a constant to be set.

As suggested by [21] for a batch sparse LVQ method, sparsity could be achieved by selecting the first Q most important coefficients, where Q is a fixed integer. However, in order to allow for more flexibility in the expression

of the prototypes, the most important coefficients are selected according to their value, by fixing a threshold: let $0 < \nu \leq 1$ be the selected threshold. At time step t at which an update occurs and for every $u = 1, ..., U$, the coefficients of the prototype p_u are first ordered in descending order, $\beta_{u,(1)} \geq ... \geq \beta_{u,(\sharp I_u(t))}$. Then, the integer N_u such that

$$N_u = \arg \min_{k=1,...,\sharp I_u(t)} \left\{ \sum_{i=1}^{k} \beta_{u,(i)} \geq \nu \right\}$$

is introduced. The most important coefficients are finally updated as follows

$$\beta_{u,(i)} = \begin{cases} \frac{\beta_{u,(i)}}{\sum_{j=1}^{N_u} \beta_{u,(j)}} & \text{if } (i) \leq N_u \\ 0 & \text{if } (i) > N_u \end{cases},$$

and $I_u(t)$ is updated accordingly afterwards by keeping the observations that correspond to non zero coefficients only.

The sparse relational SOM algorithm is entirely described in Algorithm 3.

Contrary to K-PCA SOM, in this version, the prototypes might directly be interpreted through the observations that are used in their representation. Also, the sparsity is updated during the training and the induced dimension reduction is thus not constrained to the efficiency of a given dimension reduction technique such as K-PCA. However, due to the sparse representation step, the algorithm can be computationally more expensive than K-PCA SOM and the amount of information preserved in the sparse representation is not as well controlled as in K-PCA projection. Finally, the complexity of the method, for T iterations, is not easily obtained: if a total mass equal to ν results in no more than $Q(\nu)$ observations for every prototype at each step, then the global complexity of the method has an upper bound of order $\mathcal{O}\left((Q(\nu))^2 UT\right) + \mathcal{O}\left(n(Q(\nu)^2\right)$ (respectively for the iterations and the final clustering computation). However, the relation between ν and $Q(\nu)$ is hard to know in advance and can depend on the dataset distribution and on the training itself.

4.2. Variants of the sparse updates

In the line of [21], variants of the update step (step 15 in Algorithm 3) can be introduced. More precisely, some of the approaches introduced in [21] can be used almost directly in the online framework. The main difference is

Algorithm 3 Sparse online K-SOM

For all $u = 1, \ldots, U$, initialize p_u^0 among U randomly selected observations in $(x_i)_i$. Initialize $I_u(0) = \{i(u)\}$, with $i(u) \in \{1, \ldots, n\}$ for all u and $\boldsymbol{\beta}_u^0 = 1$.
for $t = 1, \ldots, T$ **do**
    Randomly choose an input x_i, $i \in \{1, ..., n\}$.
    Assignment step: find the unit with the prototype closest to $\phi(x_i)$:

$$f^t(x_i) \leftarrow \arg\min_{u=1,\ldots,U} \left[(\boldsymbol{\beta}_u^{t-1})^\top \mathbf{K}_{I_u(t-1)} \boldsymbol{\beta}_u^{t-1} - 2 \sum_{j \in I_u(t-1)} \beta_{uj}^{t-1} K(x_j, x_i) \right]$$

    where $\mathbf{K}_{I_u(t-1)} = (K(x_j, x_{j'}))_{j,j' \in I_u(t-1)}$.
    Representation step: $\forall\, u = 1, \ldots, U$
    if $i \in I_u(t-1)$, **then**
        $\boldsymbol{\beta}_u^t \leftarrow \boldsymbol{\beta}_u^{t-1} + \mu(t) H^t(d(f^t(x_i), u)) \left(\mathbf{1}_i - \boldsymbol{\beta}_u^{t-1} \right)$
        $I_u(t) = I_u(t-1)$
    else if $i \notin I_u(t-1)$, **then**

$$\boldsymbol{\beta}_u^t \leftarrow \left[1 - \mu(t) H^t(d(f^t(x_i), u))\right] \left(\boldsymbol{\beta}_u^{t-1}, 0\right) + \mu(t) H^t(d(f^t(x_i), u)) (\underbrace{0, ..., 0}_{\sharp I_u(t-1)}, 1)$$

        $I_u(t) = I_u(t-1) \cup \{i\}$.
    end if
    Sparse representation
    if t is an update instant **then**
        Sparsely update the prototypes: $\forall\, u = 1, \ldots, U$ and with $\beta_{u,(1)}^t \geq ... \geq \beta_{u,(\sharp I_u(t))}^t$, set

$$N_{t,u} = \arg\min_{k=1,\ldots,\sharp I_u(t)} \left\{ \sum_{i=1}^{k} \beta_{u,(i)}^t \geq \nu \right\}$$

    and $\forall\,(i)$, st $i \in I_u(t)$,

$$\beta_{u,(i)}^t \leftarrow \begin{cases} \frac{\beta_{u,(i)}^t}{\sum_{j=1}^{N_{t,u}} \beta_{u,(j)}^t} & \text{if } (i) \leq N_{t,u} \\ 0 & \text{if } (i) > N_{t,u} \end{cases}$$

    and $I_u(t) \leftarrow \{i : (i) \leq N_{t,u}\}$.
    end if
end for

that the number of iterations in the online version is much larger than in the batch version. In practice, this implies that the sparse approximation must have a low computational complexity because it is performed several times during the training. We thus restrict ourselves to the following two proposals which are the less computationally costly and can be used instead of step 15 in Algorithm 3:

- *simple numerical heuristic approximation* (N-**num**): instead of selecting the first coefficients which sum to a given amount of total mass, one can simply select the first N coefficients for a given N. This changes step 15 into

$$\beta^t_{u,(i)} = \begin{cases} \frac{\beta^t_{u,(i)}}{\sum_{j=1}^{N} \beta^t_{u,(j)}} & \text{if } (i) \leq N \\ 0 & \text{if } (i) > N \end{cases},$$

- *simple geometric heuristic approximation* (N-**geom**): as an alternative, one can search through the observations in I_u the first N observations closest to prototype p_u. Additionally to what is described in [21], we set the coefficients $\beta_{u,i}$ afterwards, in accordance to the new observations used to represent the prototype p_u. Again, we used the previous values of the coefficients as *a priori* for the new ones. Step 15 re-writes:

  1. $\tilde{I}_u(t)$ is the set of the first N observations, x_i, in $I_u(t)$ which minimize $\|p_u - \phi(x_i)\|^2$
  2. $\boldsymbol{\beta}^t_{u,i} \leftarrow \begin{cases} \frac{\beta^t_{u,i}}{\sum_{j \in \tilde{I}_u(t)} \beta^t_{u,j}} & \text{if } (i) \in \tilde{I}_u(t) \\ 0 & \text{if } i \notin \tilde{I}_u(t) \end{cases},$
  3. $I_u(t) \leftarrow \tilde{I}_u(t)$

Both approaches described above lead to a fixed number of observations for representing all the prototypes, contrary to what is proposed in Section 4.1. The advantage of our proposal is that prototypes may be represented by a number of observations which varies with the density of the data on the map: when a given cluster is isolated from its neighbors, this should yield to a more accurate way to represent the cluster. Also, for all three methods, the sparse approximation is maintained during all the training, as long as the sparse updates are performed frequently enough. This yields to a reduced computational cost of the assignment step (Step 4 in Algorithm 3) and, to a lesser extent, of the representation step (Step 10 in Algorithm 3).

As already stated in the previous section, the computational gain is not easily obtained and is a trade-off between the sparsity constraint imposed on the data (which itself depends on the total mass ν or on the number of observations N and on the number of update instants) and the complexity of the sparse representation step itself (Step 15 in Algorithm 3). Further discussions about computational time and efficiency of the different alternatives are provided in the experiments (Section 6).

5. The case of dissimilarity data

For the sake of clarity, the entire paper has been written in the framework of kernel data. However, its extension to dissimilarity data is almost straightforward, even in the case where the dissimilarity is not Euclidean. The present section briefly describes the modifications that must be made in this setting.

In the sequel, Δ will denote a dissimilarity matrix with entries $\delta(x_i, x_j)$, the dissimilarity between the observations x_i and x_j. Δ is supposed to have some very basic properties, such as the positiveness of all entries ($\delta(x_i, x_j) \geq 0$), symmetry ($\delta(x_i, x_j) = \delta(x_j, x_i)$) and a null diagonal $\delta(x_i, x_i) = 0$. However, it may not necessarily be Euclidean (see, for instance [18] for a discussion on the additional requirements which make this dissimilarity Euclidean). The extension of SOM to this case is called "relational SOM" and was described in [19, 18]. It is very similar to the kernel case, except that the assignment step of Algorithm 1 is replaced by

$$f^t(x_i) \leftarrow \arg\min_{u=1,\dots,U} \left[\left(\beta_u^{t-1}\right)^\top \boldsymbol{\Delta}_i - \frac{1}{2} \left(\beta_u^{t-1}\right)^\top \Delta \beta_u^{t-1} \right], \tag{7}$$

where $\boldsymbol{\Delta}_i$ is the i-th row in matrix Δ. This equation is justified by the pseudo-Euclidean framework described in [15]: Δ can be expressed as the difference between dot products of images of the original data by two mapping functions, ψ_1 and ψ_2 into two Euclidean spaces, $\mathcal{E}_1$ and $\mathcal{E}_2$:

$$\delta(x_i, x_j) = \|\psi_1(x_i) - \psi_1(x_j)\|_{\mathcal{E}_1} - \|\psi_2(x_i) - \psi_2(x_j)\|_{\mathcal{E}_2} .$$

In this framework, Equation (7) is the exact distance calculation in the space $\mathcal{E}_1 \otimes \mathcal{E}_2$ equipped with the pseudo dot product $\langle ., . \rangle_{\mathcal{E}_1} - \langle ., . \rangle_{\mathcal{E}_2}$. Moreover, if

Δ is Euclidean, then the similarity defined by

$$K(x_i, x_j) = -\frac{1}{2}\left(\delta(x_i, x_j) - \frac{1}{n}\sum_{k=1}^{n}(\delta(x_i, x_k) + \delta(x_k, x_j)) + \frac{1}{n^2}\sum_{k,\,k'=1}^{n}\delta(x_k, x_{k'})\right), \tag{8}$$

as suggested in [36], is a kernel and the distances between observations in the feature space induced by this kernel are given by Δ. Thus, Equation (7) is exactly equivalent to the assignment step in standard SOM for the kernel K defined by Equation (8).

Hence, sparse K-SOM can straightforwardly be extended to dissimilarity data using the assignment step restricted to selected observations in the sparse representation as given in Equation (7). For SOM based on dimension reduction, the extension is also easy to obtain. For a non-Euclidean dissimilarity, Δ, the eigendecomposition of the similarity K defined by Equation (8) leads to positive and negative eigenvalues (because, in this case, K is not a kernel anymore). If $n_+ \leq n$ denotes the number of positive eigenvalues, the projection of the original data by K-PCA should be restricted to the first p components associated with the p largest eigenvalues such that $p \leq n_+$. This restriction is equivalent to performing K-PCA on a kernel pre-processed with the standard "clip" approach as suggested in [47].

6. Experimental results

6.1. Methodology

The present section is devoted to the evaluation of the proposed methods on several datasets and from several points of view. First, in Section 6.2, the methods presented in this article are assessed on both small and large datasets. This allowed us to compare the performances and computational efficiency of these methods with those of the standard K-SOM algorithm, in an extensive way. Finally, in Section 6.3, the two new methods are used to produce a map on a very large dataset. In Section 6.3, computational times are reported and compared, the influence of the size of the prototypes is assessed and the Nyström approximation is tested. This example also serves as a use case to show how the results can be interpreted.

The chosen datasets were chosen in the framework of this article, *i.e.*, most of them are not standard numeric datasets. They include graphs, genomic sequences and categorical time series to illustrate the approach on a

wide range of application and type of (dis)similarities. For every method and every set of parameters that we tested, the experiments were performed with 100 maps, so as to evaluate the variability of our conclusions.

Finally, performances are reported in terms of:

- standard quality measures used to evaluate SOM (see [48] for a description of quality measures in SOM): i) the *quantization error* (QE), $\sum_{u=1}^{U} \sum_{i:\, x_i \in \mathcal{C}_u} \|\phi(x_i) - p_u\|^2$ with $\|\phi(x_i) - p_u\|^2$ given by the kernel as in Equation (3) (K-PCA SOM) or as in Equation (6) (sparse K-SOM). This measure is a clustering quality measure, disregarding the map topology; ii) the *topographic error* (TE) which is the simplest of the topographic preservation measures: it computes the ratio over $(x_i)_i$ of the second best matching units that are in the direct neighborhood of the best matching units on the map.
  Since for the K-PCA SOM version, the distances $\|\phi(x_i) - p_u\|^2$ depend on the number of selected axes (the smaller the number of axes, the smaller the distances between observations in the corresponding projected subspace), we have normalized all QE by the average of the squared distances between all pairs of observations in the feature space, $\|\phi(x_i) - \phi(x_j)\|^2$ (for $i \neq j$) to leverage the impact of the feature space metric itself. However, the performances remain difficult to compare directly even with such a normalization, as shown in the results of the simulations. Thus, the *average intra-cluster inertia* (ICI) is also provided. This measure is equal to the average over the units u of $\sum_{i:\, x_i \in \mathcal{C}_u} \|\phi(x_i) - G_{\mathcal{C}_u}\|^2$ in which $G_{\mathcal{C}_u} = \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} \phi(x_i)$ is the center of gravity of $\mathcal{C}_u$. We can easily show that it is equal to $\frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} K(x_i, x_i) - \frac{1}{(\sharp \mathcal{C}_u)^2} \sum_{i,i':\, x_i, x_{i'} \in \mathcal{C}_u} K(x_i, x_{i'})$, for kernel data and to $\frac{1}{2(\sharp \mathcal{C}_u)^2} \sum_{i,i':\, x_i, x_{i'} \in \mathcal{C}_u} \delta(x_i, x_j)$ for dissimilarity data (see Appendix Appendix A for a proof);

- quality measures which take advantage of a prior external information: i) if an *a priori* classification of the observations is given, we use the *normalized mutual information* (NMI, [49]) between this *a priori* classification and the clustering induced by the SOM map. A perfect match between the two would correspond to an NMI equal to 1 whereas independent classifications would provide an NMI equal to 0; ii) when the studied dataset is a graph and no *a priori* classification is given, the quality of the clustering of the SOM map can nevertheless be evaluated

by computing its *modularity* [50]. A high modularity indicates a good clustering with respect to the graph structure;

- quality measures which aim at providing an indication of how much the results are stable between two runs of the algorithm. To do so, we provide the average NMI between any pair of clustering results obtained from the 100 runs, for a given method and with a given set of parameters. In the sequel, this measure is called *stability*.

6.2. Evaluation of the different sparse approaches on various datasets

6.2.1. Description of the datasets

This section aims at evaluating and comparing the two approaches described in Sections 3 and 4 on various datasets, two small datasets, for an intensive analysis, and three larger datasets, for a better evaluation of the gain in computational time.

More precisely, the two small datasets are used for the experiments presented in this section:

- a graph that gives the frequencies of co-appearance in a same chapter between characters in the novel "Les Misérables" from the French author Victor Hugo. The 77 vertices of the graph are the characters and the 254 edges are weighted by the number of co-appearances in a same chapter. For this graph, a kernel obtained from the shortest path lengths is computed[1]. The similarity described in Section 5 is used to perform K-PCA SOM: in this case, the similarity K defined in Equation (8) is not positive (only the first 67 components are positive);

- a dataset that contains 465 input data issued from ten unbalanced sampled species of Amazonian butterflies. This dataset was previously used by [52] to demonstrate the synergy between DNA barcoding and morphological-diversity studies. The notion of DNA barcoding comprises a wide family of molecular and bioinformatics methods aimed at identifying biological specimens and assigning them to a species. DNA barcoding data are composed of sequences of nucleotides, *i.e.*, sequences of "A", "C", "G", "T" letters in high dimension (hundreds

[1]The graph as well as the shortest path lengths are included in the R package **SOMbrero** [51].

or thousands of sites). Specific distances and dissimilarities such as the Kimura-2P [53] are usually computed and were used in the experiments. A similarity was obtained from the Kimura-2P dissimilarity as described in Equation (8) to perform K-PCA. Again, this similarity is not positive. Only 246 eigenvalues are positive in the eigendecomposition of K.

These two datasets will be denoted, respectively, by "lesmis" and "astraptes" in the sequel.

Additionally, three larger datasets are also used for comparison:

- a graph those edges model hyper-links between blogs on US politics, recorded in 2005 by [54][2]. The 1,222 vertices of the graph represent the political blogs and are labeled by the political leaning (liberal or conservative). The graph contains 19,089 edges. The similarity used for this graph is again obtained from Equation (8), using the shortest path lengths as the original dissimilarity measure. The undirected version of the shortest path length was used to provide a symmetric dissimilarity, even though the original graph is directed. Only the first 779 eigenvalues of this similarity are positive;

- a DNA dataset similar to the "astraptes" data, except with a larger size. This dataset contains 614 sites of the CoI locus for 2,036 samples issued from the cowries family. The data were introduced in the DNA barcoding context in [55]. In order to assess the ability for clustering of the proposed algorithms, the species with very few observations were removed. The final dataset used for simulations contained 1,414 samples and 47 species, the number of observations per species varying from 11 to 140 observations. The species were used as *a priori* classes for computing the NMI. Here also, the Kimura-2P dissimilarity [53] was used for deriving the similarity matrix, for which only the first 476 eigenvalues were positive.

- a (standard) numerical dataset related to red variants of the Portuguese "Vinho Verde" wine [56][3]. The dataset contains 4 898 observations of

[2]The graph is available at `http://www-personal.umich.edu/~mejn/netdata/polblogs.zip`.

[3]The dataset is available at `https://archive.ics.uci.edu/ml/datasets/Wine+Quality`.

12 numeric variables based on physicochemical tests, such as the pH, the sulphates or the residual sugar. Additionally, the quality (a score between 0 and 10) of the wines is provided, which is used as an *a priori* class to compute the quality measure (NMI) described in Section 6.1 (and thus it is not used to compute the similarity). To stuck to the framework of the article, the Gaussian kernel was used to obtain a measure of similarity between wines: $K_{ij} = e^{-\sigma \|x_i - x_j\|^2}$ with σ equal to the median of $\left\{ \frac{1}{\|x_i - x_j\|^2} \right\}_{i<j}$.

In the sequel, these datasets will be refered as "polblogs", "cowrie" and "wines", respectively.

6.2.2. Evaluation of K-PCA SOM and direct sparse K-SOM

More precisely, the following methods are compared on these two datasets:

- a standard kernel SOM, K-SOM (or relational SOM if the dissimilarity is not Euclidean);
- K-PCA SOM, as described in Section 3. The datasets used in this section are not large enough to allow a relevant use of the Nyström approximation of the K-PCA;
- direct sparse K-SOM (or relational SOM) as described in Section 4, and denoted by sparse K-SOM in the sequel.

All methods were trained for a 5×5 grid ("lesmis"), a 8×8 grid ("astraptes") and a 10×10 grid ("polblogs", "cowrie" and "wines") with respectively 500 ("lesmis"), 2500 ("astraptes"), 6000 ("polblogs"), 7000 ("cowrie") and 8000 ("wines") iterations. These choices approximately correspond to default parameters in **SOMbrero**. All grids were equipped with a piecewise linear neighborhood with the distance between units calculated as the Euclidean distance between the unit coordinates in $\mathbb{N}^{*2}$.

Furthermore, different sets of parameters, corresponding to different levels of sparsity, were tested:

- for K-PCA SOM, the only parameter to set was the dimension of the projection, p. This parameter was chosen with respect to a minimal ratio of entropy preserved in the projection and this ratio itself was varied in {20%, 40%, 60%, 80%};

- for the direct sparse K-SOM, two parameters had to be set: the first one is the mass parameter ν which was varied in $\{90\%, 99\%\}$ for the two smallest datasets ("lesmis" and "astraptes") and in $\{95\%, 99\%\}$ for the three largest datasets ("polblogs", "cowrie" and "wines") to ensure sparse results. The second parameter calibrated the update instants: random ascending updates were performed with $\kappa \in \{1, 50\}$ for both datasets.

Tables 1-5 provide the results obtained over 100 maps (mean and standard deviation) for different quality criteria. For the smallest datasets, we focused on providing many different measures of the quality of the resulting map, as compared to the one obtained by the standard K-SOM: normalized QE, ICI, TE and modularity (lesmis) or NMI (astraptes), as described in Section 6.1, are given and the last column of the results provides the final dimension of the prototypes (either the exact dimension for K-PCA SOM or the average over all prototypes for the sparse K-SOM). The computational time is not reported because, for these small datasets it is not relevant.

For the three largest datasets, the computational time is reported but the results are less exhaustive on quality criteria (only the most important ones are reported: ICI, TE, modularity/NMI and stability). Also, for K-PCA SOM, only the clustering time is given. The computational time for the K-PCA itself is $\sim$3.73 seconds for "polblogs", $\sim$3.73 seconds for "cowrie" and $\sim$5.65 seconds for "wines", which is less than the computational time needed to train the map in all cases.

6.2.3. Discussion on the comparison

Results demonstrate a high efficiency of both the DR method and the sparse approach to decrease data dimensionality while producing accurate results in a reasonable computational time. For almost all quality criteria, both approaches introduced in the manuscript are in the range of or outperform the results obtained with the direct K-SOM at a very reduced computational time and a limited dimensionality of the resulting prototypes.

More specifically, QE is always better for K-PCA SOM with the smallest dimensionality. However, this is merely an effect of the dimensionality of the input data and is not a reliable criterion to compare results. ICI is a better way to measure the cluster homogeneity and shows that in almost all cases, K-PCA SOM and sparse K-SOM have comparable or even better ICI than the original approach (direct sparse SOM) with a slight advantage

Table 1: Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "lesmis" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	QE (×100)	ICI (×100)	TE (%)	Modularity (×100)	Stability (%)	Dimension
K-SOM	23.40 (0.44)	47.57 (2.53)	3.01 (2.52)	31.76 (2.80)	85.04 (2.13)	77
K-PCA (80%)	18.30 (0.46)	47.31 (2.51)	3.30 (3.01)	32.02 (3.27)	86.20 (2.21)	27
K-PCA (60%)	10.32 (0.55)	48.09 (2.41)	3.36 (3.65)	**32.17** (2.63)	89.56 (2.17)	11
K-PCA (40%)	2.55 (0.25)	52.62 (2.42)	**2.64** (3.30)	25.96 (1.77)	91.85 (2.08)	4
K-PCA (20%)	**0.92** (0.18)	52.35 (2.21)	2.83 (3.32)	24.53 (1.92)	**92.68** (2.02)	2
sparse (90%, 1)	29.34 (1.20)	48.80 (2.88)	9.64 (7.00)	31.44 (4.09)	77.72 (2.94)	4
sparse (90%, 50)	23.25 (5.61)	**46.63** (2.70)	4.45 (3.62)	30.96 (3.03)	83.64 (2.34)	8
sparse (99%, 1)	23.36 (4.06)	**46.63** (2.71)	3.25 (2.85)	31.40 (3.03)	84.64 (2.18)	13
sparse (99%, 50)	23.40 (4.49)	47.07 (2.57)	3.31 (2.72)	31.74 (3.41)	84.99 (2.35)	15

Table 2: Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "astraptes" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	QE (×100)	ICI ($\times 10^4$)	TE (%)	NMI (%)	Stability (%)	Dimension
K-SOM	1.02 (0.06)	3.37 (0.43)	**1.39** (1.99)	82.19 (0.59)	94.57 (0.88)	459
K-PCA (80%)	0.21 (0.02)	4.34 (0.70)	1.79 (2.20)	**90.73** (2.86)	**95.20** (1.05)	5
K-PCA (60%)	0.10 (0.01)	6.40 (1.42)	2.71 (2.64)	80.99 (0.67)	94.73 (0.96)	3
K-PCA (40%)	0.06 (0.01)	9.33 (1.52)	2.64 (2.27)	79.14 (0.59)	94.49 (0.95)	2
K-PCA (20%)	**0.01** (0.00)	23.15 (2.06)	24.55 (8.84)	73.02 (0.61)	94.70 (0.85)	1
sparse (90%, 1)	2.08 (1.46)	3.59 (8.97)	2.94 (3.09)	86.34 (1.24)	91.87 (1.40)	5
sparse (90%, 50)	0.99 (0.10)	**3.08** (5.70)	3.72 (4.16)	82.60 (0.66)	93.62 (0.86)	8
sparse (99%, 1)	0.98 (0.05)	3.30 (4.90)	1.51 (2.54)	82.22 (0.58)	94.66 (0.84)	25
sparse (99%, 50)	20.99 (0.06)	3.38 (4.55)	**1.39** (1.63)	82.16 (0.59)	94.73 (0.77)	25

for sparse K-SOM. Only, sparse K-SOM with the "wines" dataset exhibits a poor performance for this criterion.

The quality of the organization of the map is measured with TE, which

Table 3: Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "polblogs" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	ICI (×100)	TE (%)	NMI (%)	Stability (%)	CPU time	Dimension
K-SOM	84.01 (0.85)	21.93 (1.51)	20.56 (0.30)	64.81 (0.73)	18269 (2378)	1599
K-PCA (80%)	86.39 (0.84)	14.01 (1.42)	20.93 (0.27)	69.68 (0.79)	27.60 (4.45)	257
K-PCA (60%)	86.61 (0.77)	14.87 (1.35)	20.96 (0.26)	70.62 (0.82)	18.88 (2.12)	121
K-PCA (40%)	89.35 (2.28)	13.56 (2.12)	21.14 (0.32)	60.27 (1.73)	16.55 (2.45)	50
K-PCA (20%)	91.46 (0.87)	**12.06** (1.51)	**21.30** (0.25)	**77.19** (0.77)	**13.35** (2.35)	13
sparse (95%, 1)	85.20 (1.40)	34.03 (2.80)	20.64 (0.41)	55.14 (0.41)	60.13 (2.50)	8
sparse (95%, 50)	**84.00** (0.89)	32.61 (2.31)	20.66 (0.32)	60.39 (0.32)	67.89 (7.86)	12
sparse (99%, 1)	84.18 (0.75)	23.95 (1.66)	20.46 (0.30)	63.98 (0.30)	88.19 (5.30)	34
sparse (99%, 50)	84.06 (0.76)	24.15 (1.62)	20.51 (0.30)	64.30 (0.30)	96.73 (6.75)	34

Table 4: Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "cowrie" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	ICI (×100)	TE (%)	NMI (%)	Stability (%)	CPU time	Dimension
K-SOM	2.07 (0.24)	1.57 (1.29)	**90.45** (0.86)	**94.79** (1.04)	3771 (1534)	1414
K-PCA (80%)	2.05 (0.19)	1.92 (1.39)	90.26 (0.73)	94.55 (0.89)	16.68 (1.45)	23
K-PCA (60%)	1.94 (0.21)	2.33 (1.33)	89.11 (0.81)	94.04 (0.80)	15.19 (1.31)	10
K-PCA (40%)	3.15 (0.30)	3.85 (1.96)	85.72 (0.80)	90.49 (0.93)	15.35 (2.41)	4
K-PCA (20%)	6.26 (0.28)	5.11 (2.18)	74.47 (0.48)	87.99 (1.17)	**15.04** (2.27)	2
sparse (95%, 1)	2.09 (0.26)	2.04 (1.31)	89.64 (0.97)	91.78 (1.18)	70.03 (1.23)	8
sparse (95%, 50)	**1.84** (0.20)	2.38 (1.82)	90.37 (0.70)	93.78 (0.92)	62.66 (3.27)	13
sparse (99%, 1)	20.36 (0.22)	1.55 (1.36)	90.34 (0.79)	94.45 (0.91)	90.73 (4.16)	37
sparse (99%, 50)	20.00 (0.22)	**1.54** (1.24)	90.37 (0.79)	94.50 (0.93)	81.14 (2.89)	37

is often very comparable in both approaches to the direct K-SOM. However, it is again poor for the sparse K-SOM applied to the "wines" dataset and frequently tends to increase when ICI decreases. A good compromise between

Table 5: Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "wines" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	ICI (×100)	TE (%)	NMI (%)	Stability (%)	CPU time	Dimension
K-SOM	22.10 (0.50)	10.37 (1.34)	**11.86** (0.31)	74.23 (0.81)	13480 (10575)	1222
K-PCA (80%)	**21.94** (0.51)	10.15 (1.31)	11.78 (0.30)	74.96 (0.77)	20.00 (3.29)	28
K-PCA (60%)	22.43 (0.61)	8.80 (1.30)	11.72 (0.34)	75.37 (0.88)	17.95 (1.74)	9
K-PCA (40%)	25.44 (0.74)	7.08 (1.33)	**11.86** (0.26)	77.65 (0.88)	17.32 (2.33)	4
K-PCA (20%)	34.99 (0.36)	**0.13** (0.72)	11.01 (0.23)	**85.75** (1.68)	**15.99** (1.88)	2
sparse (95%, 1)	47.62 (1.46)	14.29 (1.35)	11.75 (0.36)	66.21 (0.79)	81.48 (5.02)	8
sparse (95%, 50)	45.90 (1.08)	11.95 (1.39)	11.76 (0.38)	70.72 (0.77)	97.60 (14.28)	13
sparse (99%, 1)	44.14 (0.95)	11.26 (1.36)	11.80 (0.30)	74.00 (0.73)	141.17 (12.28)	40
sparse (99%, 50)	44.17 (0.93)	11.58 (1.24)	11.83 (0.32)	74.13 (0.77)	146.93 (14.98)	39

ICI and TE is reached for K-PCA SOM with an preserved entropy rate of 60% and for sparse K-SOM with ν equal to 90%/95% and κ equal to 50 for all datasets.

When comparing the different results with the *a priori* external information (through modularity or NMI), again the proposed approaches are comparable to or even better than the direct sparse K-SOM. However, in some cases ("lesmis", "astraptes", "cowrie"), it tends to highly deteriorate when the dimensionality is decreasing. This shows that a good tradeoff has to be found between interpretability (small dimensionality) and preservation of most of the information from the original dataset.

Finally, the stability of the results is often increased by K-PCA SOM (except for the "cowrie" dataset). This is an expected behavior: since the dimensionality of the input dataset is reduced, the prototypes lies in a reduced dimensionality space with less degrees of freedom. On the contrary, sparse K-SOM is not as appealing for this criterion because, even if the prototype representation is constrained, it is expressed in the original data space, which has a high dimension.

These good performance of our approaches come along with a high computational efficiency: K-PCA SOM and sparse K-SOM allows to highly reduce the computational times. Even if the K-PCA itself is a problem with a high

complexity, it does not affect much the efficiency of K-PCA SOM on large datasets with a few thousands observations. The reason is that relational SOM is itself a time consuming algorithm, as explained in the introduction of this article. For a more challenging dataset (with more than ten thousands observations), a comparison and discussion is provided in Section 6.3.

In conclusion, both approaches introduced in this article are valid alternative to the direct K-SOM for large relational datasets. They both provide simplified prototype representation at a very reduced computational time. The dimensionality of the final prototypes is in all our examples 10 times or even 100 times smaller than in the direct sparse SOM version with no loss in accuracy of the resulting map. However, it should be noted that the easiness to interpret the prototypes is not exactly equivalent in K-PCA SOM and in sparse K-SOM: in the first approach, prototypes are expressed on the axes of the K-PCA which have to be interpreted in a previous step of the analysis, whereas, in sparse K-SOM, they are directly related to (a few number of) original observations. A detailed study is then presented in Sections 6.3.2 and 6.3.3.

6.2.4. Comparison with other approaches

In this section, the datasets "lesmis" and "astraptes" presented in Section 6.2.1 are used to compare the sparse approach to the simple numerical heuristic approximation (denoted N-num) and the simple geometric heuristic approximation (denoted N-geom) described in Section 4.2. Both strategies lead to a fixed number of observations for representing all the prototypes. Thus, for a fair comparison, this number was chosen considering the average final dimensionality of the prototypes obtained with the sparse method (last column in Tables 1 and 2). The random ascending updates are performed using the same setting than in the original sparse version (*i.e.*, random ascending updates $\kappa \in \{1, 50\}$).

Results obtained with N-num and N-geom over 100 maps are provided in Table 6 for "lesmis" and in Table 7 for "astraptes". In average, for both datasets, N-num gives better results than N-geom. As observed with the sparse method, both N-num and N-geom obtain best results with a final dimension equal to 8 for "lesmis". For all quality criteria except for the ICI, our variant of the sparse updates slightly improves the heuristic approximation approaches. This conclusion is supported by the results on "astraptes", except that N-num gives a better classification than our variant of the sparse updates.

Table 6: Comparison between the different variants for the sparse updates (average over 100 maps and standard deviation between parenthesis) for the "lesmis" dataset. Parameters for the methods are given between parenthesis after the method name (maximum mass, ν, and update parameter, κ, for random ascending updates).

Methods	QE (×100)	ICI (×100)	TE (%)	Modularity (×100)	Stability (%)
N-num (4, 1)	31.36 (1.66)	50.12 (3.16)	20.72 (9.37)	31.13 (4.23)	75.31 (3.49)
N-num (8, 50)	23.35 (0.53)	46.89 (2.60)	4.71 (3.36)	31.39 (3.14)	83.62 (2.20)
N-num (13, 1)	23.45 (0.45)	46.99 (2.55)	3.38 (2.99)	31.85 (3.07)	**84.85** (2.32)
N-num (15, 50)	23.40 (0.44)	47.04 (2.28)	**3.21** (2.44)	**31.92** (3.20)	84.41 (2.41)
N-geom (4, 1)	35.09 (3.31)	51.56 (4.95)	22.31 (12.91)	25.90 (3.45)	72.14 (4.93)
N-geom (8, 50)	23.76 (0.81)	**45.00** (2.66)	7.49 (4.90)	24.40 (3.06)	81.06 (2.47)
N-geom (13, 1)	25.32 (1.37)	46.26 (2.39)	5.48 (4.08)	27.27 (2.54)	82.10 (2.40)
N-geom (15, 50)	**23.28** (0.53)	46.45 (2.56)	4.77 (3.60)	29.12 (2.89)	83.43 (2.14)

Table 7: Comparison between the different variants for the sparse updates (average over 100 maps and standard deviation between parenthesis) for the "astraptes" dataset. Parameters for the methods are given between parenthesis after the method name (maximum mass, ν, and update parameter, κ, for random ascending updates).

Methods	QE (×100)	ICI ($\times 10^4$)	TE (%)	NMI (%)	Stability (%)
N-num (5, 1)	3.27 (0.82)	3.80 (0.90)	6.56 (5.47)	**88.08** (1.56)	90.88 (1.98)
N-num (8, 50)	1.25 (0.13)	**3.28** (0.62)	2.88 (3.33)	83.51 (0.79)	93.14 (0.98)
N-num (25, 1)	1.01 (0.72)	3.36 (0.49)	1.99 (2.80)	82.20 (0.65)	94.42 (0.90)
N-num (25, 50)	**1.00** (0.07)	3.34 (0.48)	**1.71** (2.37)	82.10 (0.59)	**94.44** (0.82)
N-geom (5, 1)	28.44 (12.16)	27.08 (11.37)	35.95 (14.65)	77.44 (6.99)	75.30 (6.72)
N-geom (8, 50)	2.23 (1.36)	5.82 (1.13)	11.45 (6.99)	83.77 (1.22)	92.71 (1.21)
N-geom (25, 1)	29.86 (7.45)	20.12 (7.96)	22.00 (9.73)	81.47 (2.51)	88.51 (2.90)
N-geom (25, 50)	1.74 (0.96)	5.01 (0.79)	8.72 (5.93)	83.19 (0.79)	93.84 (1.08)

For the same level of sparsity, the results obtained with both N-num and N-geom slightly deteriorate the map quality compared to what can be observed with our version of the sparse updates.

6.3. Using K-PCA SOM and sparse K-SOM for mining job trajectories

This section presents the experiments performed on a more realistic dataset, with a larger sample size, obtained from the survey "Generation 98" [11, 57]. The dataset contains information on career paths of 16,040 young people monitored during 94 months after having graduated in 1998. Nine categories are used to describe labor market statuses: permanent-labor contract, fixed-term contract, apprenticeship contract, public temporary-labor contract, on-call contract, unemployed, inactive, military service, education. Dissimilarities between career trajectories were computed using the optimal matching [58, 7] on the 12,560 unique career paths. This resulted in a non positive dissimilarity (6,651 eigenvalues out of 12 500 were found positive).

To assess the accuracy and the computational cost of both K-PCA SOM and sparse SOM, 100 maps were trained, using an R implementation of the methods, on the same 40-nodes computer without concurrent access. All maps were trained for a 10×10 grid, equipped with a piecewise linear neighborhood, with 60,000 iterations. The entropy ratio preserved by K-PCA SOM was varied in {20%, 40%, 60%, 80%}. For sparse K-SOM, the mass parameter ν was varied in {95%, 99%} and the update parameter κ was varied in {1, 50}. Only 10 maps were trained using the standard K-SOM due to its very high computational cost (more than ten days for each map).

Table 8 presents the results obtained in terms of QE, ICI, TE and CPU time (only the clustering time is reported, the K-PCA computational time is $\sim 8,000$ seconds. A detailed study to address this issue is made in Section 6.3.4). The last column provides the final dimension or number of coefficients of the prototypes.

As observed in Section 6.2.3, results demonstrate a high efficiency, in term of computational cost, of both K-PCA SOM and sparse K-SOM, while producing accurate results. Direct K-SOM takes more than ten days to train one map, whereas the slowest alternative strategy only requires sixteen minutes (*i.e.*, the sparse K-SOM with $\nu = 99\%$ and $\kappa = 1$). The results also confirm what was found with the toy datasets: K-PCA SOM provides a good trade-off between a good map quality and low dimensional prototypes and outperforms sparse K-SOM. The best results for K-PCA SOM are obtained with 20% entropy-rate preserved. However, this strategy selects only two dimensions, which increases the redundancy in the data and tends to produce clusters with few observations. Thus, the K-PCA SOM preserving 40%

Table 8: Performance results of K-PCA SOM and sparse K-SOM (average over 100 maps and standard deviation between parenthesis) for the "trajectories" dataset. Parameters for the methods are given between parenthesis after the method name (% of entropy preserved in the projection for K-PCA SOM and maximum mass, ν, and update parameter, κ, for random ascending updates in sparse K-SOM).

Methods	QE (×100)	ICI	TE (%)	CPU time	Stability (%)	Dimension
K-SOM	20.99 (0.12)	**23.94** (0.24)	7.91 (0.66)	949582 (1 373)	77.65 (3.66)	12 500
K-PCA (80%)	23.49 (0.10)	24.07 (0.31)	8.27 (0.92)	251 (78)	75.81 (1.82)	392
K-PCA (60%)	15.36 (0.12)	24.32 (0.32)	8.57 (0.77)	136 (44)	75.72 (1.95)	44
K-PCA (40%)	5.61 (0.09)	26.26 (0.37)	6.98 (0.75)	114 (40)	77.13 (3.24)	8
K-PCA (20%)	**0.37** (0.00)	31.92 (0.95)	**0.82** (0.86)	**112** (33)	**86.31** (5.69)	2
sparse (95%, 1)	32.40 (0.74)	28.66 (1.36)	30.86 (6.88)	378 (3)	55.79 (1.37)	14
sparse (95%, 50)	25.36 (0.35)	26.57 (0.59)	11.15 (1.86)	655 (32)	63.64 (0.88)	14
sparse (99%, 1)	25.11 (0.17)	27.09 (0.46)	5.26 (0.72)	1025 (194)	68.08 (1.25)	50
sparse (99%, 50)	26.76 (0.36)	27.36 (0.71)	31.59 (4.53)	381 (28)	59.81 (0.90)	8

entropy should be preferred. The best results for the sparse K-SOM are obtained with a mass equal to 95%.

Both K-PCA SOM and sparse K-SOM provide accurate results in a reasonable computational time. For sparse K-SOM, prototypes can be interpreted by inspecting the properties of the few observations used to represent them. For K-PCA SOM, the projection of the data on a subspace requires to interpret the axes of the K-PCA as an extra step in order to understand the meaning of the prototypes. A detailed study showing how the results of both approaches can be interpreted is performed in Sections 6.3.2 and 6.3.3.

6.3.1. Analysis of the influence of the prototype size

The job trajectory dataset was further used to assess the influence of the size of the prototypes on different characteristics. More precisely, we conducted an experiment to analyze the relation between the dimension of the projection (in K-PCA SOM) or the average number of coefficients per prototype (in sparse K-SOM) and some numerical characteristics of the algorithm: ICI, TE, CPU time and stability. A larger set of parameters, corresponding to varying dimensions of the prototypes, were tested by training 20 maps with each value of the parameters:

- for K-PCA SOM, the number of dimensions used for the projection

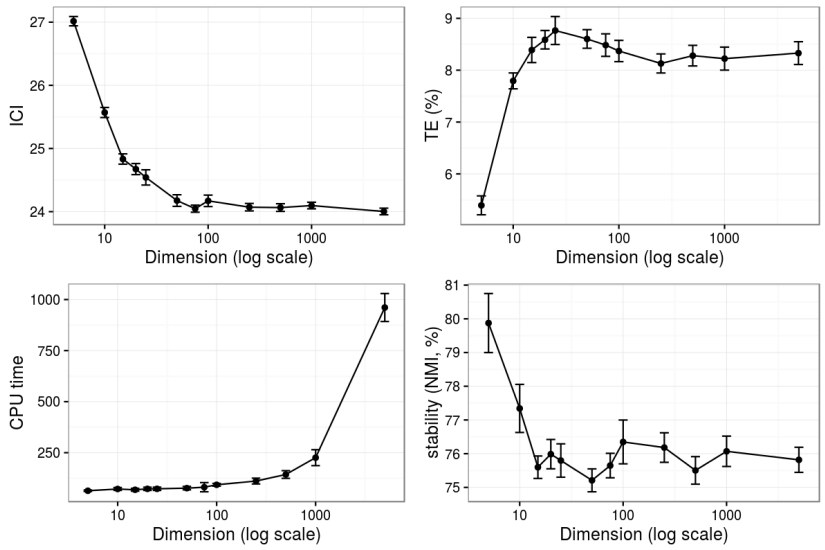

Figure 1: **K-PCA SOM**. Average performances over 20 maps (ICI, TE, computational time in seconds and stability as measured by NMI) versus the dimensionality. Error bars correspond to the standard error $\frac{\text{standard deviation}}{\sqrt{20}}$.

was varied in $\{5, 10, 15, 20, 25, 50, 75, 100, 250, 500, 1000, 5000\}$. Figure 1 displays the evolution of different numerical characteristics of the maps versus the projection dimension;

- for sparse K-SOM, following the results given in Table 8, we set the random ascending update parameter κ to 1 and varied the maximum mass, ν in $\{0.9, 0.95, 0.975, 0.99, 0.995\}$. Figure 2 provides the evolution of different numerical characteristics of the maps versus the average number of coefficients per prototype.

For K-PCA SOM, up to approximately 50 (over a maximum of $12,500$), the dimension seems to have only a very limited effect on the quality of the map. All quality criteria stabilize after this value, with a slight tendency to improve and then to deteriorate again for very high dimensions. A strong computational time benefit can be observed when decreasing the dimensionality below $1,000$: this is a direct consequence of the quadratic complexity of K-SOM.

For sparse K-SOM, all characteristics, except for CPU time, tend to improve when the number of coefficients increases. However, TE seems to sta-

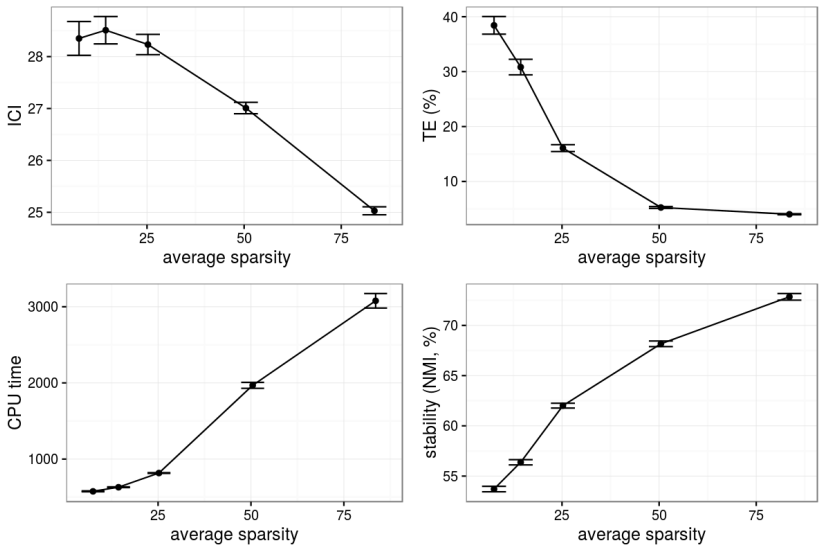

Figure 2: **Sparse K-SOM**. Average performances over 20 maps (ICI, TE, computational time in seconds and stability as measured by NMI) versus the average sparsity (average number of coefficients per prototypes) for different values of ν. Error bars correspond to the standard error $\frac{\text{standard deviation}}{\sqrt{20}}$.

bilize for approximately 50 coefficients per prototypes ($\nu = 0.99$). However, since the computational time is not reduced from the same amount than in K-PCA SOM, testing the value of ν is not a good strategy. Moreover, $\nu = 0.995$ gives prototypes with approximately 80 coefficients, which must not be increased to preserve interpretability.

6.3.2. Interpretability of K-PCA SOM

In this section, the map with the lowest ICI, among the 100 generated by the K-PCA SOM with 40% preserved entropy rate, is used to show how the results of K-PCA SOM can be interpreted, despite the K-PCA preprocessing. Its performances are equal to 5.74 (QE), 25.42 (ICI) and 7.66 (TE). The projection of the data on a subspace requires to interpret the K-PCA axes first. Figure 3 (left) presents the entropy supported by the first 15 axes and shows that the first two axes are enough to provide relevant information on the data. Figure 3 (right) displays the projections of the observations on the first two principal axes. The first axis represents 16.90% of the total entropy and opposes permanent-labor and fixed term contracts. This is supported by Figure 4 which shows the distribution of the 25 career

paths with the smallest and the highest coordinates on the first two axes of the K-PCA. Stable job trajectories have the smallest coordinates on the first axis when fixed-term contract or unemployed have the highest. Figure 3 (right) also demonstrates that the second axis separates two kinds of precarious situations. Fixed-term contracts are opposed to highly precarious contracts such as unemployment, inactivity, on-call contract and public temporary-labor contract. The same observations can be made regarding the first 25 career paths with the smallest and highest coordinates on the second K-PCA axis, as shown in Figure 4.

The distribution of the job trajectories within each neuron of the map is represented in Figure 5. First note that the presented map is comparable in term of topology to the one described in [18]. Different typologies can be highlighted: a fast access to permanent contracts (clear blue) on the bottom-left corner of the map, a transition through fixed-term contracts before obtaining stable ones (dark and then clear blue) on the map top-left corner, temporary jobs (dark blue) on the top-middle neurons, a long period of inactivity (yellow) or unemployment (red) on the map bottom-right corner.

The map organization is in accordance with the axis interpretation. Figure 6 (top) displays the average coordinates on the first and second axes in every cluster of the map. Results show a gradient of the observation coordinates on the first K-PCA axis between the bottom-left and the right side of the map. This gradient can also be seen on the map shown in Figure 5 and on the heatmaps presented on Figure 6 (bottom) which represent the cluster average of the career path modes[4]. This confirms that the first principal component (and corresponding diagonal on the map) separates permanent contracts from instable career paths. In Figure 6 (top), a gradient can also be observed for the second K-PCA axis between the top-left, where trajectories correspond to a fast access to permanent-labor contracts and the bottom-left corner of the map, where trajectories pertaining to precarious jobs are gathered.

6.3.3. Interpretability of sparse K-SOM

Similarly to the previous section, the present section provides a short discussion about one of the final results obtained from sparse K-SOM. The

[4]To compute mode averages, job market contracts have been converted to numerical labels from 1, for the permanent-labor contract, to 9 for education.

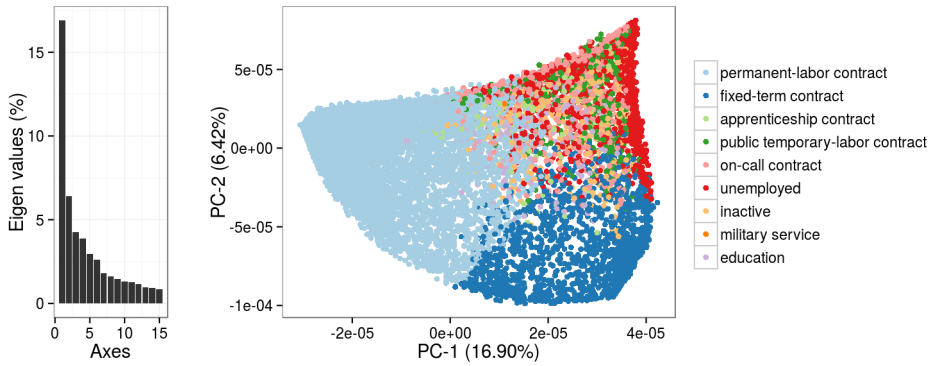

Figure 3: Entropy preserved by the 15 first axes on the left and projection of the observations on the first two principal components on the right. Colors represent the contract that appears the most often (mode) in the trajectory.

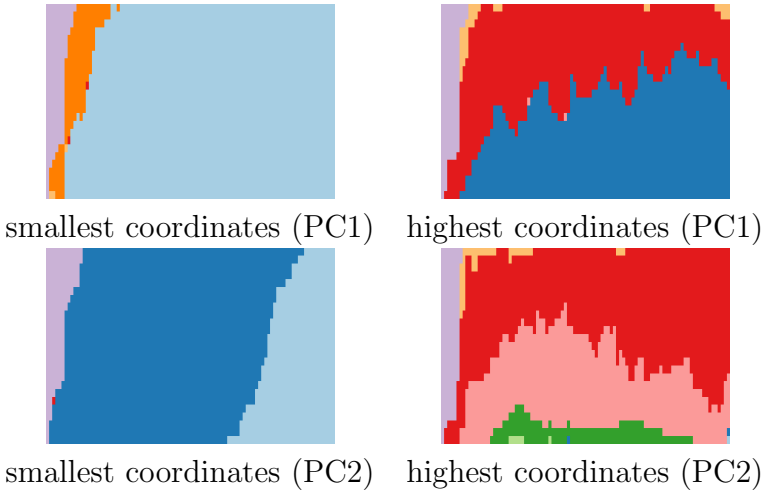

Figure 4: Distribution of the 25 career paths with smallest and highest coordinates on the first two axes of the K-PCA.

selected map is again the one with the smallest ICI among all maps obtained with $\nu = 95\%$ and $\kappa = 50$. It gives better performances in term of ICI (24.87) than K-PCA SOM but QE (25.27) and TE (10.8) are increased.

The resulting distribution of the job trajectories within the clusters of the map is provided in Figure 7. This distribution is fairly similar to the one obtained in Section 6.3.2: the left hand side of the map corresponds to a fast access to permanent contracts whereas the right hand side corresponds to

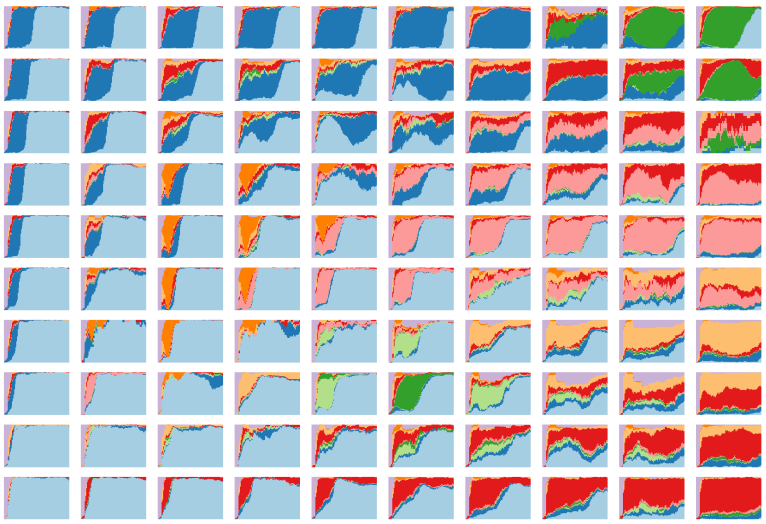

Figure 5: **K-PCA SOM**. For each neuron of the map, job trajectories distribution is represented using the observations classified in the corresponding unit. Colors represent the type of contract.

different types of precarious situations. Two main differences can be highlighted: first, the class are more homogeneous in sparse K-SOM, especially at the border of the map. This is a direct effect of the dimension reduction in K-PCA SOM: since the dimension reduction increases redundancy in the dataset, some clusters (mostly located at the borders of the map) contain more observations and are thus less homogeneous. Second, the precarious situations (on the right hand side of the map) are organized a bit differently (with on-call contracts in the middle or the bottom of the map). However, both representations are realistic, with most of the clusters in the map being homogeneous.

A similar representation is provided in Figure 8 (left) but restricted to the observations which are involved in the prototype definition. The right part of this figure displays the number of such observations. Two conclusions can be derived from these graphics: the first one is that the observations involved in the prototype definition have a distribution very similar to the distribution of the entire set of observations included in the corresponding cluster. They are thus a selected subset of observations representative of their cluster. Moreover, as their number is very restricted compared to the total number of observations included in a cluster (approximately 14/15 observations as

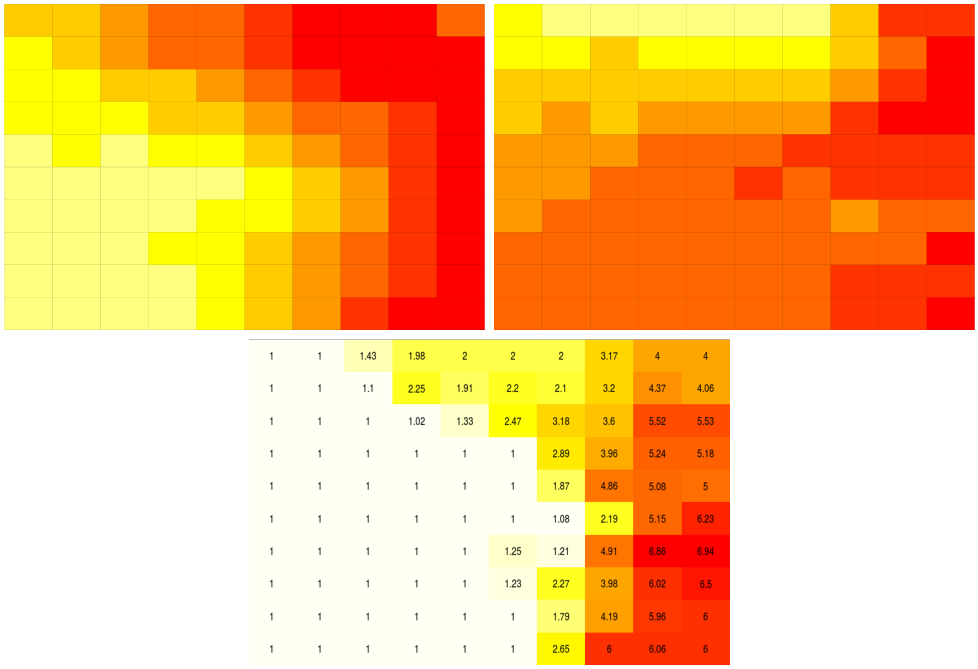

Figure 6: **K-PCA SOM**. Representation of the SOM map with neurons filled using colors according to the average coordinate of the observations for the first (on the top-left) and the second (on the top-right) principal component. On the bottom, the map neurons are filled using colors according to the average of the career path modes.

shown in the left part of Figure 8), they are a convenient way for the user to make sense of the prototypes and thus, of the corresponding cluster, since an exhaustive inspection of these observations becomes possible.

6.3.4. Nyström approximation

To evaluate the relevance of using a Nyström approximation of the K-PCA, the K-PCA SOM with 40% preserved entropy rate is used as a reference. Table 9 presents K-PCA SOM results using a Nyström approximation with different rates of observations sampled to perform the approximation. This rate was varied in {100%, 25%, 10%, 5%, 1%}, in which the 100%-results are reported from Table 8. The coefficients given to the K-PCA SOM are restricted to the first eight K-PCA axes everywhere, to avoid any bias related to data dimensionality. The computational time is reported in Table 9 and gives the time needed to perform the K-PCA only, excluding the training and clustering times.

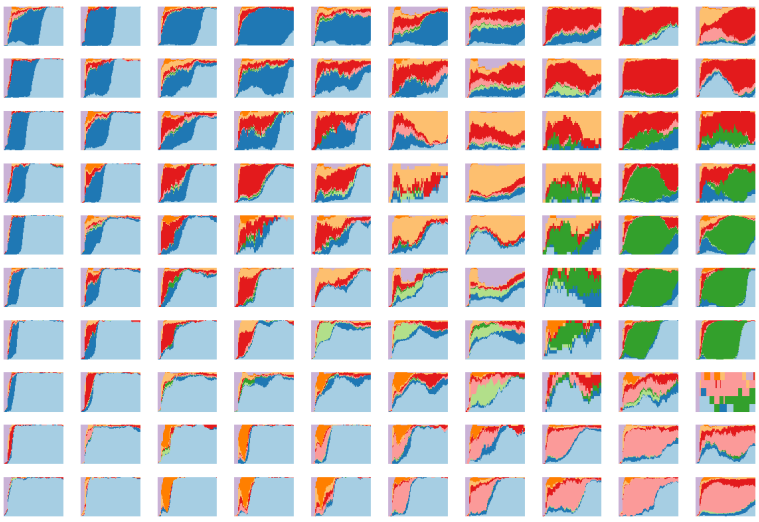

Figure 7: **Sparse K-SOM**. For each neuron of the map, job trajectories distribution is represented using the observations classified in the corresponding unit. Colors represent the type of contract.

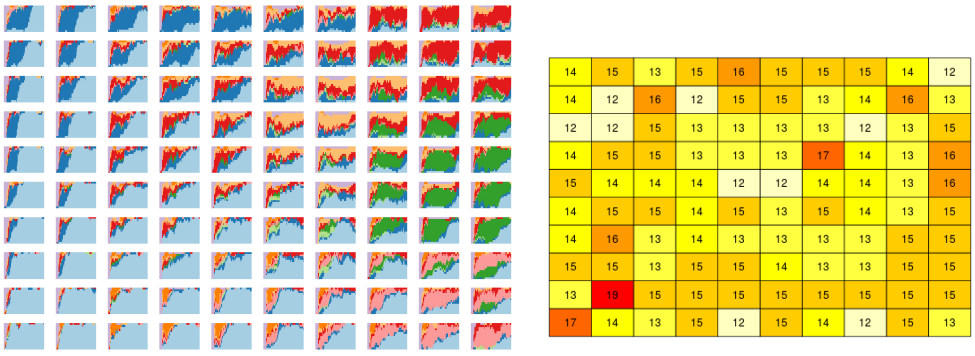

Figure 8: **Sparse K-SOM**. For each neuron of the map, job trajectories distribution for the observations with a positive coefficients for the corresponding neuron (left) and number of such observations (right).

Results demonstrate a high efficiency in terms of computational time of the Nyström approximation while producing accurate results. In fact, none of the tested values lead to deteriorate the map quality in term of QE, ICI and TE, while the K-PCA is $\sim$ 1000 times faster when using 10% of observations. The best ICI is even obtained using only 1% of the observations. The clustering stability decreases with the number of observations used by

the Nyström approximation, even if the stability is still high when using at least 10% of the observations.

The maps with the smallest ICI among the 100 maps generated from a Nyström approximation using 1% and 5% of the observations are displayed in Figure 9. Results show the ability of the Nyström approximation to preserve a realistic representation of the dataset while reducing the computational time.

The map obtained with 5% of the observations shows an organization similar to the one presented in Section 6.3.2, except for one fact: precarious situations, located on the right side of the map, are organized differently. Unemployment and public temporary-labor contracts are inverted between the top and the bottom of the map. With a Nyström approximation using 1% of the observations, trajectories mostly containing fixed-term contracts are located on the right hand side of the map, what is not observed on the map obtained with 5% of the observations and on maps presented in Section 6.3.2 and in Section 6.3.3. As expected, clusters obtained using 1% of the observations are less homogeneous than those obtained with 5%. The differences between these two maps might have different causes. Firstly, the instability of the SOM algorithm can explain the differences in terms of map organization: different runs of the algorithm give different results. This is particularly critical when the dataset to be analyzed is high dimensional as can be the "Generation 98" survey (even with a subsampling rate of 1%, the dimensionality of the problem is still larger than 100). This issue could be addressed by aggregating strategies, as described in [59]. Secondly, the differences between the two maps in Figure 9 might be explained by the high redundancy of trajectories with fixed-term contracts in the dataset: a very small subsampling might enforce the over-representation of these trajectories and affect the result. Such a problem could be addressed by using more efficient sampling techniques such as the ones described in [45].

The choice of the ratio m/n of observations to select in order to obtain accurate results highly depends on the quality of the kernel approximation provided by the Nyström technique. This quality is strongly influenced by the rank of the kernel, which can not easily be obtained when n is very large. Adaptative sampling technique for the Nyström algorithm, such as the one described in [60, 61] are based on an unequal probability sampling, which is performed iteratively and depends on the reconstruction error. [45] proposes an improved version in which the full kernel is not even needed to estimate the reconstruction error. Such methods could be relevant to assess

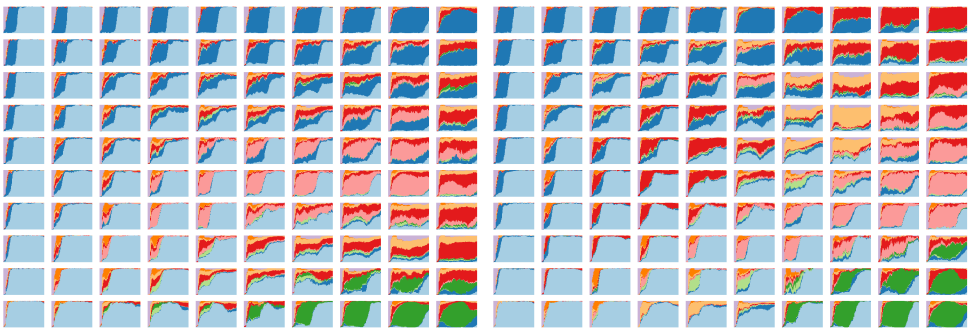

Figure 9: K-PCA SOM performed through a Nyström approximation using 1% (left) and 5% (right) of the observations: For each neuron of the map, job trajectories distribution is represented using the observations classified in the corresponding unit. Colors represent the type of contract.

Table 9: Performance results of the K-PCA SOM with K-PCA performed through a Nyström approximation (average over 100 maps and standard deviations between parenthesis) for the "trajectories" dataset. After the method name and between parenthesis, the percentage of observations used to perform the approximation is given.

Methods	QE (×100)	ICI	TE (%)	CPU time	Stability (%)
K-PCA (100%)	**5.61** (0.09)	26.26 (0.37)	**6.98** (0.75)	8 153 (205)	**77.13** (3.24)
K-PCA (25%)	5.62 (0.09)	26.11 (0.40)	7.12 (0.77)	101.38 (18.96)	75.84 (2.38)
K-PCA (10%)	5.62 (0.13)	26.13 (0.40)	7.00 (0.76)	7.39 (1.23)	74.68 (2.25)
K-PCA (5%)	5.64 (0.15)	26.05 (0.45)	7.11 (0.92)	0.86 (0.38)	73.10 (1.72)
K-PCA (1%)	5.65 (0.18)	**25.99** (0.47)	7.02 (1.02)	**0.02** (0.01)	69.32 (1.26)

the evolution of the quality reconstruction in a growing sample and to stop the Nyström sampling when this quality is considered good enough.

7. Conclusion

The contributions of the present manuscript to the analysis of (dis)similarity data with topographic maps are twofolds: firstly, we have proposed a new version of the kernel and relational SOM algorithms, called sparse K-SOM, which ensures a sparse representation of the prototypes. Secondly, this approach has been compared to a preprocessing of the data by a dimension-reduction technique (K-PCA). We have also investigating the use

of a Nyström approximation technique to ensure a better scalability of the method.

The experiments performed on several real datasets showed that both presented methods allow to strongly decrease the overall computational time on large datasets. The interpretability of the results is also improved since the prototypes have a much lower dimensionality. Moreover, the accuracy of the final map, in terms of cluster homogeneity, quality of the organization or adequacy to external *a priori* information is not deteriorated.

In average, K-PCA SOM gives better results in term of map and clustering quality than the sparse approach. However, prototypes returned by sparse K-SOM can be directly interpreted by inspecting the properties of the few observations used to represent them. In K-PCA SOM, the axes of the K-PCA have to be interpreted as an extra step to understand the prototypes meaning: this step is fairly standard in K-PCA. In conclusion, when selecting one or the other method, the user should also consider his/her need to easily interpret the result.

With the introduction of these methods, a further step is taken in allowing relational SOM to deal with massive data sets. Future works should investigate the multiple relational SOM, presented in [18], with a view to integrate several sources of data of different types, while preserving the interpretability of the results.

8. Acknowledgements

We thank the two anonymous reviewers and the editor for valuable comments and suggestions which helped to improve the quality of the paper. Part of this work was carried out using the resources from the INRA GENOTOUL bioinformatic platform `http://bioinfo.genotoul.fr`. We are grateful to the platform staff and especially to Marie-Stéphane Trotard and Didier Laborie.

References

[1] T. Kohonen, Self-Organizing Maps, 3rd Edition, Vol. 30, Springer, Berlin, Heidelberg, New York, 2001.

[2] B. Penn, Using self-organizing maps to visualize high-dimensional data, Computers & Geosciences 31 (5) (2005) 531–544.

[3] M. Pölzlbauer, M. Dittenbach, A. Rauber, Advanced visualization of self-organizing maps with vector fields, Neural Networks 19 (6-7) (2006) 911–922, advances in Self Organising Maps - WSOM'05.

[4] P. Sarlin, S. Rönnqvist, Cluster coloring of the self-organizing map: an information visualization perspective, in: 18th International Conference on Information Visualisation, IEEE, London, UK, 2013, pp. 532–538.

[5] A. Neme, J. Pulido, M. noz A., S. Hernández, T. Dey, Stylistics analysis and authorship attribution algorithms based on self-organizing maps, Neurocomputing 147 (2015) 147–159, advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012).

[6] Z. Yu, H. Wong, J. You, G. Han, Visual query processing for efficient image retrieval using a SOM-based filter-refinement scheme, Information Science 203 (2012) 83–101.

[7] A. Abbott, J. Forrest, Optimal matching methods for historical sequences, Journal of Interdisciplinary History 16 (1986) 471–494.

[8] C. Elzinga, Sequence similarity: a nonaligning technique, Sociological Methods and Research 32 (3-29).

[9] C. Lozupone, M. Hamady, S. Kelley, R. Knight, Quantitative and qualitative β eiversity measures lead to different insights into factors that structure microbial communities, Applied and Environmental Microbiology (2007) 1576–1585.

[10] Z. Yu, J. You, L. Li, H. Wong, G. Han, Representative distance: a new similarity measure for class discovery from gene expression data, IEEE Transactions on NanoBioscience 11 (4) (2012) 341–351.

[11] M. Cottrell, P. Letrémy, How to use the Kohonen algorithm to simultaneously analyse individuals in a survey, Neurocomputing 63 (2005) 193–207.

[12] T. Kohohen, P. Somervuo, Self-organizing maps of symbol strings, Neurocomputing 21 (1998) 19–30.

[13] B. Conan-Guez, F. Rossi, A. El Golli, Fast algorithm and implementation of dissimilarity self-organizing maps, Neural Networks 19 (6-7) (2006) 855–863.

[14] N. Aronszajn, Theory of reproducing kernels, Transactions of the American Mathematical Society 68 (3) (1950) 337–404.

[15] L. Goldfarb, A unified approach to pattern recognition, Pattern Recognition 17 (5) (1984) 575–582.

[16] D. Mac Donald, C. Fyfe, The kernel self organising map., in: Proceedings of 4th International Conference on knowledge-based Intelligence Engineering Systems and Applied Technologies, 2000, pp. 317–320.

[17] R. Boulet, B. Jouve, F. Rossi, N. Villa, Batch kernel SOM and related Laplacian methods for social network analysis, Neurocomputing 71 (7-9) (2008) 1257–1273.

[18] M. Olteanu, N. Villa-Vialaneix, On-line relational and multiple relational SOM, Neurocomputing 147 (2015) 15–30.

[19] B. Hammer, A. Hasenfuss, Topographic mapping of large dissimilarity data sets, Neural Computation 22 (9) (2010) 2229–2284.

[20] F. Rossi, How many dissimilarity/kernel self organizing map variants do we need?, in: T. Villmann, F. Schleif, M. Kaden, M. Lange (Eds.), Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2014), Vol. 295 of Advances in Intelligent Systems and Computing, Springer Verlag, Berlin, Heidelberg, Mittweida, Germany, 2014, pp. 3–23.

[21] D. Hofmann, F. Schleif, B. Paaß en, B. Hammer, Learning interpretable kernelized prototype-based models, Neurocomputing 141 (2014) 84–96.

[22] C. Chu, S. Kim, Y. Lin, Y. Yu, G. Bradski, A. Ng, K. Olukotun, Map-Reduce for machine learning on multicore, in: J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, A. Culotta (Eds.), Advances in Neural Information Processing Systems (NIPS 2010), Vol. 23, Hyatt Regency, Vancouver, Canada, 2010, pp. 281–288.

[23] X. Chen, M. Xie, A split-and-conquer approach for analysis of extraordinarily large data, Statistica Sinica 24 (2014) 1655–1684.

[24] S. del Rio, V. López, J. Beniítez, F. Herrera, On the use of MapReduce for imbalanced big data using random forest, Information Sciences 285 (2014) 112–137.

[25] M. Bădoiu, S. Har-Peled, P. Indyk, Approximate clustering via coresets, in: J. Reif (Ed.), Proceedings of the 34th annual ACM Symposium on Theory of Computing, no. 250-257, ACM New York, NY, USA, Montreal, QC, Canada, 2002.

[26] D. Yan, L. Huang, M. Jordan, Fast approximate spectral clustering, in: J. Elder, F. Soulié-Fogelman, P. Flach, M. Zaki (Eds.), Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining, ACM New York, NY, USA, 2009, pp. 907–916.

[27] A. Kleiner, A. Talwalkar, P. Sarkar, M. Jordan, A scalable bootstrap for massive data, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 76 (4) (2014) 795–816.

[28] N. Laptev, K. Zeng, C. Zaniolo, Early accurate results for advanced analytics on mapreduce, in: Proceedings of the 28th International Conference on Very Large Data Bases, Vol. 5 of Proceedings of the VLDB Endowment, Istanbul, Turkey, 2012.

[29] X. Meng, Scalable simple random sampling and stratified sampling, in: Proceedings of the 30th International Conference on Machine Learning (ICML 2013), Vol. 28 of JMLR: W&CP, Georgia, USA, 2013.

[30] A. Saffari, C. Leistner, J. Santner, M. Godec, H. Bischof, On-line random forests, in: Proceedings of IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), IEEE, 2009, pp. 1393–1400.

[31] M. Denil, D. Matheson, N. de Freitas, Consistency of online random forests, in: Proceedings of the 30th International Conference on Machine Learning (ICML 2013), 2013, pp. 1256–1264.

[32] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: T. Leen, T. Dietterich, V. Tresp (Eds.), Advances in

Neural Information Processing Systems (Proceedings of NIPS 2000), Vol. 13, Neural Information Processing Systems Foundation, Denver, CO, USA, 2000.

[33] R. Hochking, The analysis and selection of variables in linear regression, Biometrics.

[34] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society, series B 58 (1) (1996) 267–288.

[35] Z. Yu, P. Luo, J. You, H. S. Wong, H. Leung, S. Wu, J. Zhang, G. Han, Incremental semi-supervised clustering ensemble for high dimensional data clustering, IEEE Transactions on Knowledge and Data Engineering 28 (3) (2016) 701–714.

[36] J. Lee, M. Verleysen, Nonlinear Dimensionality Reduction, Information Science and Statistics, Springer, New York; London, 2007.

[37] C. Bouveyron, C. Brunet-Saumard, Model-based clustering of high-dimensional data: a review, Computational Statistics & Data Analysis 71 (2014) 52–78.

[38] F. Rossi, A. Hasenfuss, B. Hammer, Accelerating relational clustering algorithms with sparse prototype representation, in: Proceedings of the 6th Workshop on Self-Organizing Maps (WSOM 07), Neuroinformatics Group, Bielefeld University, Bielefeld, Germany, 2007.

[39] D. Hofmann, A. Gisbrecht, B. Hammer, Efficient approximations of robust soft learning vector quantization for non-vectorial data, Neurocomputing 147 (2015) 96–106.

[40] A. Gisbrecht, B. Mokbel, B. Hammer, The Nyström approximation for relational generative topographic mappings, in: NIPS workshop on challenges of Data Visualization, Whistler BC, Canada, 2010.

[41] X. Zhu, A. Gisbrecht, F. Schleif, B. Hammer, Approximation techniques for clustering dissimilarity data, Neurocomputing 90 (2012) 72–84.

[42] A. Gisbrecht, A. Shultz, B. Hammer, Parametric nonlinear dimensionality reduction using kernel t-SNE, Neurocomputing 147 (2015) 71–82.

[43] J. Mariette, M. Olteanu, J. Boelaert, N. Villa-Vialaneix, Bagged kernel SOM, in: T. Villmann, F. Schleif, M. Kaden, M. Lange (Eds.), Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2014), Vol. 295 of Advances in Intelligent Systems and Computing, Springer Verlag, Berlin, Heidelberg, Mittweida, Germany, 2014, pp. 45–54.

[44] B. Schölkopf, A. Smola, K. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (5) (1998) 1299–1319.

[45] S. Kumar, M. Mohri, A. Talwalkar, Sampling techniques for the Nyström method, Journal of Machine Learning Research 13 (2012) 981–1006.

[46] M. Olteanu, N. Villa-Vialaneix, Sparse online self-organizing maps for large relational data, in: E. Merényi, M. Mendenhall, O. P. (Eds.), Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2016), Vol. 428 of Advances in Intelligent Systems and Computing, Springer International Publishing Switzerland, Houston, TX, USA, 2016, pp. 27–37.

[47] Y. Chen, E. Garcia, M. Gupta, A. Rahimi, L. Cazzanti, Similarity-based classification: concepts and algorithm, Journal of Machine Learning Research 10 (2009) 747–776.

[48] G. Pölzlbauer, Survey and comparison of quality measures for self-organizing maps, in: J. Paralic, G. Polzlbauer, A. Rauber (Eds.), Proceedings of the Fifth Workshop on Data Analysis (WDA'04), Elfa Academic Press, Sliezsky dom, Vysoke Tatry, Slovakia, 2004, pp. 67–82.

[49] L. Danon, A. Diaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, Journal of Statistical Mechanics (2005) P09008.

[50] M. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical Review, E 69 (2004) 026113.

[51] J. Boelaert, L. Bendhaïba, M. Olteanu, N. Villa-Vialaneix, SOMbrero: an r package for numeric and non-numeric self-organizing maps, in: T. Villmann, F. Schleif, M. Kaden, M. Lange (Eds.), Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of

WSOM 2014), Vol. 295 of Advances in Intelligent Systems and Computing, Springer Verlag, Berlin, Heidelberg, Mittweida, Germany, 2014, pp. 219–228.

[52] P. Hebert, E. Penton, J. Burns, D. Janzen, W. Hallwachs, Ten species in one: DNA barcoding reveals cryptic species in the neotropical skipper butterfly astraptes fulgerator, Genetic Analysis 101 (41) (2004) 14812–14817.

[53] M. Kimura, A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences, Journal of Molecular Evolution 16 (1980) 111–120.

[54] L. Adamic, N. Glance, The political blogosphere and the 2004 us election: divided they blog, in: Proceedings of the 3rd LINKDD Workshop, ACM Press, New York, NY, USA, 2005, pp. 36–43.

[55] C. Meyer, G. Paulay, DNA barcoding: error rates based on comprehensive sampling, PLoS Biology 3 (12).

[56] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decision Support Systems 47 (4) (2009) 547–553.

[57] E. Côme, M. Cottrell, P. Gaubert, Analysis of professional trajectories using disconnected self-organizing maps, Neurocomputing 147 (2015) 185–196, advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012).

[58] S. Needleman, C. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, Journal of Molecular Biology 48 (3) (1970) 443–453.

[59] J. Mariette, N. Villa-Vialaneix, Aggregating self-organizing maps with topology preservation, in: E. Merényi, M. Mendenhall, O. P. (Eds.), Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2016), Vol. 428 of Advances in Intelligent Systems and Computing, Springer International Publishing Switzerland, Houston, TX, USA, 2016, pp. 27–37.

[60] P. Drineas, M. Mahoney, S. Muthukrishnan, Relative-error CUR matrix decompositions, SIAM Journal on Matrix Analysis and Applications 30 (2) (2008) 844–881.

[61] A. Gittens, M. Mahoney, Revisiting the nystrom method for improved large-scale machine learning, Journal of Machine Learning Research 28 (3) (2013) 567–575.

Appendix A. Formula for the average intra-cluster intertia

The intra cluster inertia is the average over $u \in \{1, \ldots, U\}$ of the quantities

$$\mathcal{I}(u) = \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} \|\phi(x_i) - G_{\mathcal{C}_u}\|^2$$

in which $G_{\mathcal{C}_u} = \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} \phi(x_i)$ is the center of gravity of $\mathcal{C}_u$.

Expansion for kernel data

For kernel data, this quantity equals

$$\begin{aligned}
\mathcal{I}(u) &= \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} \left\| \phi(x_i) - \frac{1}{\sharp \mathcal{C}_u} \sum_{i':\, x_{i'} \in \mathcal{C}_u} \phi(x_{i'}) \right\|^2 \\
&= \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} \left[\|\phi(x_i)\|^2 - 2 \frac{1}{\sharp \mathcal{C}_u} \sum_{i':\, x_{i'} \in \mathcal{C}_u} \langle \phi(x_i), \phi(x_{i'}) \rangle + \left\| \frac{1}{\sharp \mathcal{C}_u} \sum_{i':\, x_{i'} \in \mathcal{C}_u} \phi(x_{i'}) \right\|^2 \right] \\
&= \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} K(x_i, x_i) - 2 \frac{1}{(\sharp \mathcal{C}_u)^2} \sum_{i,i':\, x_i, x_{i'} \in \mathcal{C}_u} K(x_i, x_{i'}) + \frac{1}{(\sharp \mathcal{C}_u)^2} \sum_{i,i':\, x_i, x_{i'} \in \mathcal{C}_u} K(x_i, x_{i'}) \\
&= \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} K(x_i, x_i) - \frac{1}{(\sharp \mathcal{C}_u)^2} \sum_{i,i':\, x_i, x_{i'} \in \mathcal{C}_u} K(x_i, x_{i'}).
\end{aligned}$$

Expansion for dissimilarity data

Using the result of Equation (2) in [18], we get that

$$\mathcal{I}(u) = \frac{1}{\sharp \mathcal{C}_u} \sum_{i:\, x_i \in \mathcal{C}_u} \left[\boldsymbol{\Delta}_i \nu_u - \frac{1}{2} \nu_u^T \Delta \nu_u \right],$$

in which $\nu_u = \frac{1}{\sharp \mathcal{C}_u} \mathbf{1}_{\mathcal{C}_u}$ where the entries of $\mathbf{1}_{\mathcal{C}_u}$ are equal to 1 for indexes i' such that $x_{i'} \in \mathcal{C}_u$ and to 0 otherwise. Thus

$$\begin{aligned}
\mathcal{I}(u) &= \frac{1}{\sharp \mathcal{C}_u} \sum_{i,i' \colon x_i, x_{i'} \in \mathcal{C}_u} \frac{1}{\sharp \mathcal{C}_u} \delta(x_i, x_{i'}) - \frac{1}{2(\sharp \mathcal{C}_u)^2} \sum_{j,j' \colon x_j, x_{j'} \in \mathcal{C}_u} \delta(x_j, x_{j'}) \\
&= \frac{1}{2\sharp(\mathcal{C}_u)^2} \sum_{i,i' \colon x_i, x_{i'} \in \mathcal{C}_u} \delta(x_i, x_{i'}).
\end{aligned}$$