Kernel and dissimilarity methods for exploratory analysis in a social context

Jérôme J. Mariette, Madalina Olteanu, Nathalie Vialaneix

Kernel and dissimilarity methods for exploratory analysis in a social context

Jérôme Mariette, Madalina Olteanu and Nathalie Vialaneix

Abstract While most of statistical methods for prediction or data mining have been built for data made of independent observations of a common set of p numerical variables, many real-world applications do not fit in this framework. A more common and general situation is the case where a relevant similarity or dissimilarity can be computed between the observations, providing a summary of their relations to each other. This setting is related to the *kernel* framework that has allowed to extend most of standard statistical supervised and unsupervised methods to any type of data for which a relevant such kernel can be obtained. The present chapter aims at presenting kernel methods in general, with a specific focus on the less studied unsupervised framework. We illustrate its usefulness by describing the extension of self-organizing maps and by proposing an approach to combine kernels in an efficient way. The overall approach is illustrated on categorical time series in a social-science context and allows to illustrate how the choice of a given type of dissimilarity or group of dissimilarities can influence the output of the exploratory analysis.

1 Introduction

While most of statistical methods for prediction or data mining have been built for data made of independent observations of a common set of p numerical variables, many real-world applications do not fit in this framework. Typical such examples include categorical variables, relations between entities (*e.g.*, a graph or network) or even more complex frameworks such as categorical time series. A particularly useful simplification of these more general situations is the case where a relevant

Jérôme Mariette and Nathalie Vialaneix
INRAE, UR875 Mathématiques et Informatique Appliquées Toulouse, F-31326 Castanet-Tolosan, France e-mail: {firstname.lastname}@inrae.fr

Madalina Olteanu
SAMM, Université Paris 1, F-75005 Paris, France e-mail: madalina.olteanu@univ-paris1.fr

similarity or dissimilarity can be computed between the observations, providing a summary of their relations to each other. In addition, when this similarity has some mild additional properties, it is called a *kernel* and provides a strong mathematical framework [5] for extending most of standard statistical supervised and unsupervised methods to any type of data for which a relevant such kernel can be obtained [12, 46]. This approach has already proven useful in computational biology [45] or in social sciences and humanities [7, 35].

Nevertheless, the choice of a relevant kernel is still an open problem. Some authors have proposed to combine all candidate kernels into a "meta kernel" which is an "optimal" linear or convex combination of the individual kernels. This approach is known as the "multiple kernel learning problem" and has been widely studied in the supervised framework [17]. The present chapter aims at presenting the less addressed unsupervised framework. More precisely, after a brief introduction to kernels and their relation with the more general similarity/dissimilarity settings (Section 2), we describe how statistical methods can be extended to the kernel framework by using the so-called "kernel trick" (Section 3). Section 4 focuses more precisely on the extension of an exploratory method, called Self-Organizing Maps [23], to the kernel framework and discusses the issue of complexity and how it can be solved in this particular setting. Section 5 explains how kernels can be combined in an unsupervised setting, as a processing prior the unsupervised methods presented before. The overall approach is illustrated in Section 6 on categorical time series in a social science context: originally developed in bioinformatics, sequence analysis is indeed increasingly used in social sciences for the study of life-course processes. In this section, we discuss how the choice of a given type of dissimilarity or group of dissimilarities influences the output of the exploratory analysis and allows to extract relevant patterns from this particular kind of data.

2 Kernels and more general proximity data

2.1 Kernels and RKHS

Kernel methods consider the case where data are described by a *kernel* obtained from a Reproducing Kernel Hilbert Space (RKHS; [5]). Usually, the sample of interest takes values in an arbitrary space, $\mathcal{X}$ that encompasses a variety of data types. This sample is then described by a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is symmetric ($\forall\, x, x' \in \mathcal{X},\ K(x, x') = K(x', x)$) and positive ($\forall\, N \in \mathbb{N},\ \forall (\alpha_i)_{i=1,\dots,N} \subset \mathbb{R}$ and $\forall\, (x_i)_{i=1,\dots,N} \subset \mathcal{X},\ \sum_{i,i'=1}^{N} \alpha_i \alpha_{i'} K(x_i, x_{i'}) \geq 0$) and is called the *kernel*. Indeed, in this case, it is known [3, 5] that there exists a unique Hilbert space $(\mathcal{H}, \langle ., . \rangle_{\mathcal{H}})$ and a unique application $\phi : \mathcal{X} \to \mathcal{H}$, such that

$$\forall\, x, x' \in \mathcal{X}, \qquad \langle \phi(x), \phi(x') \rangle_{\mathcal{H}} = K(x, x').$$

$(\mathcal{H}, \langle ., . \rangle_{\mathcal{H}})$ is the RKHS of K and is also often called *feature space*; ϕ is the *feature map* of K.

In statistics and machine learning, this framework is often used to deal with observations that are not just multidimensional vectors (*e.g.*, categorical time series or graphs, among others) or to incorporate expert knowledge in the analysis (see Examples 1 and 2 below with standard examples of kernels often used in practice). The sample $(x_i)_{i=1,\ldots,n}$ is then described by pairwise relations between observations, as measured by the kernel. This leads to the computation of the *kernel matrix* $\mathbf{K} = (k_{ii'})_{i,i'=1,\ldots,n}$, with $k_{ii'} = K(x_i, x_{i'})$, which is symmetric and semi-definite positive, by definition of the kernel K.

The idea of kernel methods is to perform standard linear statistical analyses in the feature space $(\mathcal{H}, \langle ., . \rangle_{\mathcal{H}})$. Since the only operations involved in these analyses are related to the computation of dot products and norms, the Hilbert space $\mathcal{H}$ and the feature map ϕ are usually not explicitly given but used implicitly through the kernel K instead. This principle, which we illustrate below, is called the *kernel trick*.

Example 1 Some useful kernels

Kernels in $\mathbb{R}^p$. Kernel methods are often used for standard multidimensional data to provide more flexibility and non linearity in the analyses. In these spaces, a trivial kernel is given by using the standard dot product of $\mathbb{R}^p$: $K(x, x') = (x')^\top x$, which leads to the trivial feature map $\phi = \mathrm{Id}$. The feature space is then unchanged as compared to the original space ($\mathcal{X} = \mathcal{H} = \mathbb{R}^p$) and the performed statistical analysis is thus still linear.

Among more interesting kernels for $\mathbb{R}^p$, one of the most popular is the Gaussian kernel (also called Radial Basis Function – RBF – kernel) $K_\gamma(x, x') = e^{-\gamma \|x - x'\|^2}$, which shape is controlled by a hyper-parameter $\gamma > 0$. This kernel is of special importance since it is continuous and *universal* for every compact set of $\mathcal{X}$, C (meaning that the set of all functions induced by $x \in C \to K(x, .)$ is dense in the set of all continuous functions $C \subset \mathcal{X} \to \mathbb{R}$). This property allowed Steinwart [50, 51] to demonstrate the consistency of kernel classification and regression methods in the statistical sense (when the sample size grows to infinity and in terms of convergence of the error loss to its optimum). The polynomial kernel ($K(x, x') = (1 - (x')^\top x)^\gamma$ for $\gamma > 0$) and the exponential kernel ($K(x, x') = e^{(x')^\top x}$) are also universal kernels.

Kernels on graphs. In many application fields including social sciences and biology, graphs (also called networks) are widely used to represent pairwise relations between entities (friendship, professional contacts, regulation between genes, ...). A number of kernels for graphs have been proposed to provide a similarity measure between nodes based on the graph structure. Most of them are derived from regularized version of the Laplacian of the graph [25, 47] and have been used in prediction or exploratory analyses in

biology (*e.g.*, for introducing known relations between genes [54, 39]) or in social sciences (*e.g.*, to extract information from a medieval social network [7]).

2.2 From general similarities to kernels

In practice, data are often described by similarities (or dissimilarities) that are not necessarily definite positive (see Example 2). This situation is addressed either by generalizing kernel methods to the "pseudo-Euclidean" framework [16, 37], by embedding the sample directly into a Euclidean space whose dot product ressembles the original similarity (Multidimensional Scaling –MDS– is one of these approaches [11]), or by using a proper definite kernel instead of the original indefinite similarity. In the latter case, the chosen kernel is often obtained by a simple transformation of its spectrum meant to obtain only positive eigenvalues. [9, 42] are two reviews describing the topic of general similarity learning and its relation with kernel methods.

Example 2 Some more general similarities and dissimilarities

Categorical sequences or time series. Categorical sequences are naturally used in biology to represent the DNA sequences or proteins (with the categories being the amino acids). Among the many proposals for quantifying the similarities between two sequences, edit distances (also known as "Levenshtein distances" or "optimal matching dissimilarities") [36] are one of the most famous. Their main idea is to quantify the minimum number of transformations needed to obtain a sequence from another one. A cost is associated to insertion, deletion and substitution transformations to allow a flexible customization of these dissimilarities. These measures have been increasingly used in social sciences as well, for studying life-course processes [2, 35]. Section 6 describe in further details those dissimilarities.

Dissimilarities based on phylogeny. As already mentioned, kernels and dissimilarities can also embed prior expert information in their computation. A typical example is the case where variables are the abundances of different species for which a phylogeny information (a parental information between those species) is given. Such frameworks are met when studying the biodiversity of different places or in metagenomics for instance. In these applications, data are described by vectors of counts that represent the number of times given species or Operational Taxonomic Units (OTUs) have been found for a given individual. For these data, computing a measure of

proximity between observations that accounts for the distances between the species has been shown to provide a more relevant information than the simple Euclidean distance between counts [29, 30]. Such distances include the (weighted) UniFrac distance or the generalized UniFrac distance [8].

3 Basics of statistical learning with kernels

3.1 Supervised setting

A simple example of a supervised learning method that has been extended to kernels is the ridge regression. More precisely, when given a training sample $\{(\mathbf{x}_i, y_i)_{i=1,\ldots,n}\}$ for which $\mathbf{x}_i \in \mathbb{R}^p$ and y_i is a real number, the ridge regression finds the best linear predictor for $(y_i)_i$ based on $(\mathbf{x}_i)_i$ that minimizes the squared loss plus a regularization term based on the ℓ_2 norm:

$$\boldsymbol{\beta}^* = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - \beta^\top \mathbf{x}_i\right)^2 + \lambda \|\beta\|^2 \tag{1}$$

for a given $\lambda > 0$, called regularization parameter, which is usually tuned by a cross validation approach. The solution of Equation (1) is given by:

$$\boldsymbol{\beta}^* = \left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top + \lambda \mathbb{I}_p\right)^{-1} \left(\sum_{i=1}^{n} y_i \mathbf{x_i}\right),$$

that can also be written as

$$\boldsymbol{\beta}^* = \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbb{I}_n\right)^{-1} \mathbf{X}^\top \mathbf{y},$$

with $\mathbf{y} = (y_1, \ldots, y_n)^\top$ and $\mathbf{X}$ being the $(n \times p)$-matrix with rows containing the $\mathbf{x}_i$ so that the matrix $\mathbf{X}\mathbf{X}^\top$ is the $(n \times n)$-matrix with entries the pairwise dot products $\mathbf{x}_i^\top \mathbf{x}_{i'}$ for all $i, i' = 1, \ldots, n$. In summary, the solution writes

$$\boldsymbol{\beta}^* = \sum_{i=1}^{n} \alpha_i^* \mathbf{x}_i \qquad \text{with } \boldsymbol{\alpha}^* = \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbb{I}_n\right)^{-1} \mathbf{y}. \tag{2}$$

The extension of this approach to samples $(x_i)_i$ taking values in an arbitrary space $\mathcal{X}$ through the use of kernels is called *kernel ridge regression* [41]. The idea is simply to search for a linear predictor in the feature space induced by the kernel, $\mathcal{H}$, which transforms the optimization criterion of Equation (1) into:

$$w^* = \underset{w \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \langle w, \phi(x_i) \rangle_{\mathcal{H}})^2 + \lambda \|w\|^2_{\mathcal{H}}. \tag{3}$$

The best linear predictor in $\mathcal{H}$ is thus given similarly as the solution of Equation (2) but in the feature space, replacing $\mathbf{x}_i$ by $\phi(x_i)$ and the $\mathbb{R}^p$ dot product by $\langle ., . \rangle_{\mathcal{H}}$. In particular, this means that the matrix $\mathbf{X}\mathbf{X}^\top$ is replaced by a matrix with entries equal to $\langle \phi(x_i), \phi(x_{i'}) \rangle_{\mathcal{H}}$, which, by the so-called *kernel tricks*, turns out to simply be equal to $\mathbf{K}$. We thus have that

$$w^* = \sum_{i=1}^{n} \alpha_i^* \phi(x_i) \qquad \text{with } \boldsymbol{\alpha}^* = (\mathbf{K} + \lambda \mathbb{I}_n)^{-1} \mathbf{y}.$$

This result can also be found as a consequence of the Representer Theorem [22, 43] or directly solving the dual of Equation (3):

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\alpha}^\top \mathbf{K}_i \right)^2 + \lambda \|\boldsymbol{\alpha}\|^2_{\mathbf{K}}, \tag{4}$$

in which $\mathbf{K}_i$ is the i-th row of the kernel matrix $\mathbf{K}$ and $\|.\|_{\mathbf{K}}$ is the ℓ_2 norm induced by this matrix in $\mathbb{R}^n$: $\|\boldsymbol{\alpha}\|^2_{\mathbf{K}} = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$.

Variants of this framework include Support Vector Machines (SVM, [6]), for the classification case, or ϵ-SVM, for the regression case. In both cases, the main difference with the kernel ridge regression lies in the loss function but the main principle of the approach remains identical: the Representer Theorem allows to express the solution as a linear combination of the images by ϕ of the observations and the solution is obtained by solving a dual optimization problem obtained thanks to the use of the kernel trick.

3.2 Unsupervised setting

Kernel methods have also been developed for the unsupervised setting. Among the most direct of these extensions, the generalization of PCA [44] and that of k-means [13] are probably the most known and used. They both use approaches similar to the supervised case described in the previous section, and more precisely:

- computations related to the original method (*i.e.*, standard PCA and k-means) are performed in the feature space;
- to do so, the *kernel trick* is used instead of the standard computation of dot products or norms.

Kernel PCA.

Standard PCA is often presented as the eigendecomposition of the variance/covariance matrix associated to the $(n \times p)$-matrix of sample measures, $\mathbf{X}$. Assuming without loss of generality that $\mathbf{X}$ is centered, this eigendecomposition is equivalent to the dual eigendecomposition of $\mathbf{X}\mathbf{X}^\top$, that provides the coordinates (or scores) of the projection of $\mathbf{X}$ on the different principal components. More precisely, if $\mathbf{T}$ is the $(n \times k)$ column matrix with the first k eigenvectors of $\mathbf{X}\mathbf{X}^\top$, orthogonal and with a norm equal to $\frac{1}{\sqrt{\lambda_j}}$, then the $(p \times k)$ column matrix of the principal components (orthogonal and with a norm equal to 1) is $\mathbf{X}^\top\mathbf{T}$.

Kernel PCA uses a similar approach taking advantage of the analogy between $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{K}$ and between the i-th row of $\mathbf{X}$ and $\phi(x_i)$. More precisely,

1. assuming that $\mathbf{K}$ is centered in the feature space[1], the eigendecomposition of $\mathbf{K}$ is obtained. It gives $(\lambda_j)_{j=1,\ldots,k}$, the first k eigenvalues of $\mathbf{K}$, and $(t_j)_{j=1,\ldots,k}$, the associated first k orthogonal eigenvectors with a norm equal to $\frac{1}{\sqrt{\lambda_j}}$;
2. the first k (orthogonal) principal components are thus obtained as

$$w_j = \sum_{i=1}^{n} t_{ji}\phi(x_i),$$

and have a norm equal to 1 in $\mathcal{H}$. The coordinate of $\phi(x_i)$ on the j-th principal component is thus $\langle w_j, \phi(x_i)\rangle_{\mathcal{H}} = \lambda_j t_{ji}$.

Kernel k-means.

Similarly, kernel k-means performs a standard k-means algorithm in the feature space $\mathcal{H}$. To do so, in addition to computing dot products and norms using the kernel trick, it is necessary to obtain a representation of the cluster barycenters. More precisely, if $(x_i)_{i\in C}$ are the observations assigned to a given cluster C, then, the barycenter is given by:

$$\bar{x}_C = \frac{1}{|C|}\sum_{i\in C}\phi(x_i)$$

and its (squared) distance to any other observation, x_i, in the sample is obtained by:

[1] if $\mathbf{K}$ is not centered, the centering operation is simply $\mathbf{K} - \frac{1}{n}\mathbf{1}_n\mathbf{K} - \frac{1}{n}\mathbf{K}\mathbf{1}_n + \frac{1}{n^2}\mathbf{1}_n\mathbf{K}\mathbf{1}_n$, in which $\mathbf{1}_n$ is an $n \times n$ matrix with all entries equal to 1.

$$\begin{aligned}\|\phi(x_i) - \bar{x}_C\|^2_{\mathcal{H}} &= \left\|\phi(x_i) - \frac{1}{|C|}\sum_{i'\in C}\phi(x_{i'})\right\|^2_{\mathcal{H}} \\ &= \|\phi(x_i)\|^2_{\mathcal{H}} - \frac{2}{|C|}\sum_{i'\in C}\langle\phi(x_i),\phi(x_{i'})\rangle_{\mathcal{H}} + \frac{1}{|C|^2}\sum_{i'i''\in C}\langle\phi(x_{i'}),\phi(x_{i''})\rangle_{\mathcal{H}} \\ &= k_{ii} - \frac{2}{|C|}\sum_{i'\in C}k_{ii'} + \frac{1}{|C|^2}\sum_{i',i''\in C}k_{i'i''}.\end{aligned}$$

In both situations (kernel PCA and kernel k-means), the adaptation of the algorithms to kernel data is made by their direct rewriting in the feature space. New data points, that were not previously in the feature space (principal components or barycenters) are represented by linear combinations of the images by the feature map, ϕ, of observations. In addition, distances to these new elements can be expressed in function of the kernel, using the kernel trick. These adaptations are thus very similar to the supervised case situations.

4 Kernel self-organizing maps and complexity reduction

In this section, we present an extension of kernel k-means to a more general method, which simultaneously performs clustering and dimensionality reduction for visualization, namely the *self-organizing map* (SOM) algorithm. Originally designed for unsupervised exploration of standard numerical datasets [23], the method has been extended to handle non numeric data by using approaches based on Multiple Correspondence Analysis [10] or by relying on an algorithm that represents all the clusters by a prototype chosen among the data (*median SOM*, [24]). Even if very general, the latter approach is very restrictive and generates representation issues, with associated biases in the obtained maps. In the present section, we present the extension of SOM to kernels that has been introduced by several authors for batch and online versions [31, 7] and has also been generalized to data represented by general dissimilarities (rather than kernels) [19]. The second part of this section will discuss associated complexity issues when the sample size is large, and review the different strategies that can be implemented to overcome them.

4.1 Kernel self-organizing maps

For the standard case of a dataset $(\mathbf{x}_i)_{i=1,\dots,n}$ of multidimensional observations $\mathbf{x}_i \in \mathbb{R}^p$, SOM algorithm is close to k-means algorithm, except that the clusters are organized on a map equipped with a distance, d. More precisely, a map (also sometimes called a grid) is a set of U clusters (also sometimes called units or neurons) associated to physical locations in a low dimensional space. The clusters are frequently positioned in $\mathbb{R}^2$ at coordinates $(a,b)_{a=1,\dots,A,\, b=1,\dots,B}$ with $AB = U$. Clus-

ters are related to each other using pairwise distances, that can be, for instance, the Euclidean distances between their coordinates in $\mathbb{R}^2$. In addition, every cluster, u, is summarized by a *prototype*, p_u that takes its values in the input space $\mathbb{R}^p$.

When fixing the number of neurons U, one should take into account that SOM is mainly intended as a method for nonlinear mapping and dimensionality reduction –in the sense of vector quantization– than as a method for clustering the data into a small number of clusters. Some authors [53] suggest to build very large SOMs, with a number of units U larger than the sample size, n. In this context, SOM essentially reduces to nonlinear mapping and to mining the underlying distribution of the data. A second option, which is more commonly used in practice, consists in building medium size maps that are smaller than the sample size, but still large enough to have a few input observations representing each unit [23]. This strategy is a good trade-off between mapping and clustering, and a heuristic suggests to set U close to $\sqrt{n/10}$ [55].

The method aims at assigning every observation in the dataset to one of the clusters, while minimizing the distortion of the topology between the original space (here, $\mathbb{R}^p$) and the map (as seen through the distance d). The prototypes are thus expected to be representative of the observations assigned to their cluster, as the barycenter is representative of its cluster in kernel k-means. To do so, the stochastic version of the method iterates over two steps:

- an *assignment step* in which an observation, $\mathbf{x}_i$ is randomly chosen and assigned to the unit with the closest prototype:

$$f(\mathbf{x}_i) = \underset{u=1,\ldots,U}{\operatorname{argmin}} \|\mathbf{x}_i - p_u\|^2$$

  where $f(\mathbf{x}_i)$ is the cluster to which observation $\mathbf{x}_i$ is assigned;
- a *representation step* in which all prototypes are updated according to the new assignment:

$$\forall\, u = 1, \ldots, U, \qquad p_u \leftarrow p_u + \mu H(d(f(\mathbf{x}_i), u))(\mathbf{x}_i - p_u)$$

  where $\mu > 0$ is chosen so as to vanish during the training process and H is a decreasing function, generally chosen such that $H(0) = 1$ and $\lim_{z\to+\infty} H(z) = 0$.

The method is usually initialized with a random choice of the prototypes in $\mathbb{R}^p$. Different heuristics can be used to choose T: either it is defined proportionally to the sample size (typically, $T = 5n$) or it is not fixed in advance and the algorithm stops when the iterations no longer modify the solution.

The extension of SOM to kernel data is based on the same key tools as the ones described for kernel PCA and kernel k-means, and which allow to re-write the algorithm in the feature space $\mathcal{H}$:

- the prototypes are expressed as convex combinations of the images by ϕ of the observations, and the assignment step is written in terms of coefficients related to each image $\phi(x_i)$;
- the representation step is expressed with $\mathbf{K}$ by means of the kernel trick.

The full version of the method is provided in Algorithm 1.

Algorithm 1 Stochastic kernel SOM

1: $\forall u = 1, \ldots, U$ and $\forall i = 1, \ldots, n$, random initialization of the prototypes: $p_u^1 = \sum_{i=1}^n \beta_{ui}^1 \phi(x_i)$ with $\beta_{ui}^1 \in [0,1]$ and $\sum_i \beta_{ui}^1 = 1$
2: **for** $t = 1$ to T **do**
3: Select randomly one observation $i \in \{1, \ldots, n\}$ ▷ Assignment step

$$f^{t+1}(x_i) = \underset{u=1,\ldots,U}{\operatorname{argmin}} \|\phi(x_i) - p_u^t\|_{\mathcal{H}}^2$$
$$= \underset{u=1,\ldots,U}{\operatorname{argmin}} \left(k_{ii} - 2\sum_{l=1}^n \beta_{ul}^t k_{il} + \sum_{l,l'=1}^n \beta_{ul}^t \beta_{ul'}^t k_{ll'} \right)$$

4: For all $u = 1, \ldots, U$, ▷ Representation step

$$p_u^{t+1} = p_u^t + \mu_t H^t(d(f^{t+1}(x_i), u))(\phi(x_i) - p_u^t)$$
$$\Leftrightarrow \quad \beta_u^{t+1} = \beta_u^t + \mu_t H^t(d(f^{t+1}(x_i), u))\left(\mathbf{1}_i^n - \beta_u^t\right),$$

where $\mathbf{1}_i^n$ is a vector of length n with all entries equal to 0 except for the ith, which is equal to 1.
5: **end for**
6: **return** $(p_u^{T+1})_u$ (prototypes) and $(f^{T+1}(x_i))_i$ (clustering)

4.2 Complexity of kernel SOM

Kernel methods are generally considered efficient to deal with large dimensional data (when the original space is a standard multidimensional space, $\mathbb{R}^p$, with p large) but often encounter scalability issues when the sample size, n, becomes large. As noted by [40], the complexity of kernel SOM is $O(n^2UT)$ whereas the complexity of the numeric SOM in $\mathbb{R}^p$ is $O(npUT)$. When $p \ll n$ and n is large, this cost can be very prohibitive since, typically, T is of order $O(n)$. Different strategies have been developed to overcome this difficulty, among which approximations with dimensionality reduction or sparse representations [20, 32] or exact approaches using a storage of intermediate results [33]. Other alternatives to reduce the complexity of kernel methods directly use accelerated computations of the kernel dot products (used in most kernel methods) with tiled reduction schemes on GPU, without even storing the kernel itself (see KeOps, `https://www.kernel-operations.io/keops/index.html`). They would be a practicable approach to accelerate the assignment step of kernel SOM when the full kernel matrix itself does not fit in memory but we will restrict to the simpler case where the kernel is already computed and stored, in the remaining of this section.

Low rank and sparse approximations.

The first type of approaches relies on a simpler representation of the prototypes, with a reduced number of (non zero) coefficients. [32] proposes two types of solutions. The first one is very similar to the strategies developed in [20] and uses a direct sparse approach in which an additional step is added to each iterations, aiming at thresholding the smallest coefficients $(\beta_{ui})_i$ for every prototype p_u. The second method relies on a prior step (a kernel PCA) to provide inputs that are the coordinates of the original observations on the first k principal components of the kernel PCA. The SOM algorithm then used is a simple numeric SOM with a complexity of $O(nkTU)$ with $k \ll n$. However, this prior step has a high computational cost itself: the full eigendecomposition of $\mathbf{K}$ has a computational cost of $O(n^3)$ but it can be reduced with the Nyström approximation [57]. This method allows to obtain an approximation of the eigendecomposition of $\mathbf{K}$ using an eigendecomposition of a submatrix $\mathbf{K}^{(m)}$ based on m observations chosen at random in the original sample. The eigendecomposition approximation is even exact if the rank of $\mathbf{K}$ is smaller than m. The complexity of the approach is reduced to $O(nm^2)$ where m is usually chosen $\ll n$.

Exact approaches.

Most of the complexity of the kernel SOM comes from the assignment step, which is $O(n^2U)$. Re-formulating this step and transforming it into the update of stored results, we reduced it to $O(U)$. More precisely, the assignment step writes:

$$f^{t+1}(x_i) = \underset{u=1,\dots,U}{\operatorname{argmin}} A_u^t - 2B_{ui}^t$$

with $A_u^t = \sum_{j,j'=1}^n \beta_{uj}^t \beta_{uj'}^t k_{jj'}$ and $B_{ui}^t = \sum_{j=1}^n \beta_{uj}^t k_{ij}$. Storing these quantities in memory (a vector of size U and a $(U \times n)$-matrix), the representation step reduces to an update of A^t and B^t:

$$\begin{aligned} A_u^{t+1} &= (1-\lambda_u(t))^2 A_u^t + \lambda_u(t)^2 k_{ii} + 2\lambda_u(t)(1-\lambda_u(t))B_{ui}^t \\ B_{ui'}^{t+1} &= (1-\lambda_u(t))B_{ui'}^t + \lambda_u(t)k_{i'i}, \end{aligned}$$

with $\lambda_u(t) = \mu_t H^t(d(f^{t+1}(x_i), u))$. The representation step thus has a complexity of $O(nU)$ (update of B^t) and the total complexity of the approach is reduced to $O(nUT)$. This reduction of the computational time is thus obtained at the cost of storing operations with a memory cost of $O(U)$ and $O(nU)$ for A^t and B^t respectively.

5 Combining kernels

Kernel methods have proven to be particularly efficient when data are described by multi-source and multi-type information obtained on the same n observations. In this case, each source of data, of a given particular type (numerical, graph data, factors, ...), can be passed through a kernel, K^m ($m = 1, \ldots, M$): this kernel provides the similarity information between observations, seen from the point of view of the source m. The advantage of such an approach is that it provides a common representation of the different sources that can be easily combined. A similar framework is the one where multiple kernels can be obtained from a single dataset, each capturing a specific feature. Combining these kernels avoids having to choose between them, and also benefits of the information coming from different aspects of the data. Among the combination approaches [17], one that has been widely developed is the computation of a convex combination of the M kernels into a single meta-kernel:

$$\mathbf{K}^\gamma = \sum_{m=1}^{M} \gamma_m \mathbf{K}^m, \qquad \text{st } \begin{cases} \gamma_m \geq 0, \ \forall\, m = 1, \ldots, M \\ \sum_{m=1}^{M} \gamma_m = 1 \end{cases}.$$

In the context of supervised methods, the choice of $(\gamma_m)_m$ is usually done by solving a global optimization problem that aims at minimizing a prediction loss, with respect to the parameter of a given method (SVM for instance) and to the value of $(\gamma_m)_m$ [59, 58, 21, 18]. In the unsupervised setting, choosing relevant $(\gamma_m)_m$ is harder since the objective function might not be as easily designed or because, as it is the case for kernel PCA, its joint optimization to estimate the principal components and the $(\gamma_m)_m$ is degenerate [49].

Several propositions have thus been made [28, 60, 48, 56, 34] to overcome this issue and, in the latter, we proposed two solutions that can cope with non numerical observations, contrary to the others. The first method, named **STATIS-UMKL**, is based on the STATIS method [38, 26] and aims at searching for a consensual meta-kernel. More precisely, the method searches for the kernel that is the most similar, on average, to all the kernels to be combined, $(K^m)_{m=1,\ldots,M}$:

$$\max_{\mathbf{v}} \sum_{m=1}^{M} \left\langle \mathbf{K}^{\mathbf{v}}, \frac{\mathbf{K}^m}{\|\mathbf{K}^m\|_F} \right\rangle_F \quad \text{for} \quad \mathbf{K}^{\mathbf{v}} = \sum_{m=1}^{M} v_m \mathbf{K}^m, \quad \text{and} \quad \mathbf{v} \in \mathbb{R}^M \text{ such that } \|\mathbf{v}\|^2_{\mathbb{R}^M} = 1,$$

where $\langle ., . \rangle$ and $\|.\|_F$ stand for the Frobenius dot product and norm. It is easy to show that the solution is given by the spectral decomposition of a $M \times M$-matrix, $\mathbf{C}$, such that $C_{mm'} = \frac{\langle \mathbf{K}^m, \mathbf{K}^{m'} \rangle_F}{\|\mathbf{K}^m\|_F \|\mathbf{K}^{m'}\|_F}$ and γ is thus chosen as $\frac{\mathbf{v}}{\sum_m v_m}$.

The second method first creates a proxy of the local geometry induced by each kernel K^m using a k nearest neighbor graph and the global adjacency matrix of these M graphs, $\mathbf{W}$ is then used in a global criterion. This criterion is designed to preserve at best the local geometry measured by $\mathbf{W}$ in the feature space induced by

the meta-kernel $\mathbf{K}^\gamma$:

$$\underset{\gamma\in\mathbb{R}^M}{\operatorname{argmin}} \sum_{i,i'=1}^{n} W_{ii} \left\|C_i(\gamma) - C_{i'}(\gamma)\right\|^2_{\mathbb{R}^n},$$
$$\text{st } \gamma_m \geq 0 \text{ and } \sum_{m=1}^{M} \gamma_m = 1,$$

with

$$C_i(\gamma) = \left\langle \phi^\gamma(x_i), \begin{pmatrix} \phi^\gamma(x_1) \\ \vdots \\ \phi^\gamma(x_n) \end{pmatrix} \right\rangle_{\mathcal{H}^\gamma} = \begin{pmatrix} K^\gamma(x_i, x_1) \\ \vdots \\ K^\gamma(x_i, x_n) \end{pmatrix}.$$

This problem has a sparse solution that performs a selection of the kernels (some of the entries of $(\gamma_m)_m$ are forced toward 0) because of the convexity constraint $\sum_{m=1}^{M} \gamma_m = 1$ but this can be relaxed by replacing the ℓ_1 constraint with a constraint of the ℓ_2 norm instead (the two versions are called sparse-UMKL and full-UMKL where "UMKL" stands for Unsupervised Multiple Kernel Learning).

Once the kernel is obtained, it can be used as input to kernel based algorithm, like kernel PCA, kernel k-means or kernel SOM for exploratory purpose.

6 Application

The combined-kernel SOM algorithm is illustrated on data related to school-to-work transitions, extracted from the survey "Generation 98"[2]. The dataset contains information on 16,040 young people having graduated in 1998 and monitored during 94 months after having left school. The labor-market status has nine categories, labeled as follows: permanent contract, fixed-term contract, apprenticeship, public temporary contract, on-call contract, unemployed, inactive, military service, education. The following stylized facts are highlighted by a first descriptive analysis of the data, as shown in Figure 1[3]:

- permanent contracts represent more than 20% of all status after one year and their ratio continues to increase up to 50% after three years and almost 75% after seven years;
- the ratio of fixed-term contracts is more than 20% after one year on the labor market, but it is decreasing to 15% after three years and then seems to converge to 8%;
- almost 30% of the young graduates are unemployed after one year. This ratio is decreasing and becomes constant, 10%, after the fourth year.

[2] Available thanks to Génération 1998 à 7 ans - 2005, [producer] CEREQ, [diffusion] Centre Maurice Halbwachs (CMH).

[3] The graphical illustrations were carried out using the `TraMineR` package [15].

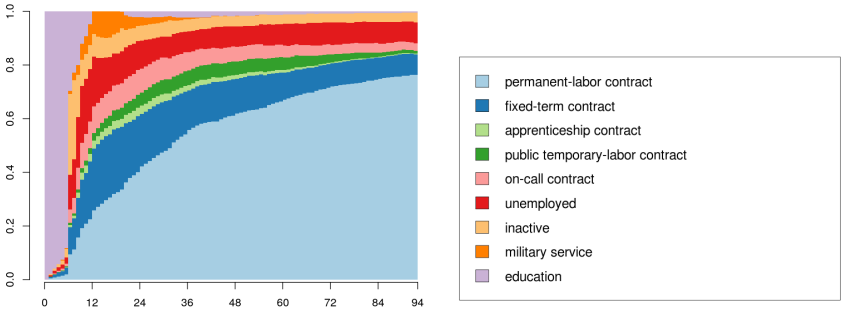

Fig. 1 Chronogram of the labor market structure as illustrated by the "Génération 98" dataset. x-axis is time (in month) and y-axis is the proportion of each type of contract.

Some trajectories were duplicated (some people had exactly the same job trajectories) so, to reduce redundancy and computational time, we used only the 12,471 unique trajectories.

Three optimal combined kernels were computed from three sets of dissimilarities with different features, and each of these kernels was then used as input to the kernel SOM. The three sets of dissimilarities are described with more details in the next section. For each set of dissimilarities, the optimal combined kernel was obtained as follows:

- first, each dissimilarity matrix, $\mathbf{D}$, was transformed into a (centered) similarity matrix by computing

$$\forall\, i,j = 1,\ldots,n,\ \ s(x_i,x_j) = -\frac{1}{2}\left(\mathbf{D}_{ij}^2 - \frac{1}{n}\sum_{l=1}^{n}\mathbf{D}_{il}^2 - \frac{1}{n}\sum_{l=1}^{n}\mathbf{D}_{jl}^2 + \frac{1}{n^2}\sum_{l,l'=1}^{n}\mathbf{D}_{ll'}^2\right).$$

  Each of these resulting similarity matrices were used as kernels, even though a small part of their spectra was non positive;
- second, the resulting kernel matrices were optimally combined using STATIS-UMKL, as described in Section 5.

All dissimilarities were computed using the R package **TraMineR** [15] and the combination of kernels was obtained using the R package **mixKernel**. Each combined kernel was processed through a kernel self-organizing map using the implementation provided in the R package **SOMbrero**. For each map to be trained, a 10×10 configuration was selected and default values of the package were chosen for the initialization step, the topology of the map (choice of H^t in Algorithm 1) and the decreasing value μ_t (also as in Algorithm 1). We decided to use 10×10 maps as a trade-off between having a meaningful visualization and a reduced number of meaningful typical trajectories. Since many trajectories are redundant –the permanent contracts are overrepresented– a map with a number of units equal to about a tenth of the number of inputs was a reasonable choice. Final results were represented

as chronograms: more precisely, each unit of the map was featured by a rectangle containing the chronogram of the subsample of trajectories assigned to this unit.

6.1 Three sets of dissimilarities and relations between them

There is currently a vast literature devoted to measuring similarities for longitudinal data, and the community agrees that the differences between the various criteria focus on three different aspects of sociological importance: the sequencing or the order in which the states appear, the timing, and the duration of the states. A recent and detailed review of these methods, focusing on these different aspects, and also introducing some new and versatile criteria, is available in [52]. Starting from these considerations, three sets of distances were selected for the present study: the first aimed at focusing on the sequencing and possibly the duration, the second on the timing, and the third on the duration only.

The first group of dissimilarities contains one criterion based on the number of matching subsequences, and two based on generalizations of the classical OM metric [36, 1]:

- The **SVRspell** distance, proposed by [14], is computed using the number of matching subsequences within the distinct sequences of states, where the durations of the spells are weighted by some parameter b. This method is built to be sensitive to sequencing, and, depending on b, it may be also made sensitive to durations. In the following, we set $b = 0$, so that the sensitivity to the order only is put forward.
- The **OMspell** distance, introduced in [52], generalizes the OM distance to sequences of spells. The increasing size of the alphabet and of the costs to be specified are controlled through a linear function depending on a parameter δ, representing the cost of extending or compressing a spell by one unit of time. For small values of δ, the method favors the expansion or the compression of existing spells. For $\delta = 0$, **OMspell** reduces to the usual OM distance between the distinct sequences of states. Let us remark here that the smaller δ, the less sensitive the criterion is to the duration of the spells and the more it is to the sequencing.
- The **OMstran** distance, also introduced in [52], adapts the OM distance to sequences of transitions. Here also, the alphabet and the number of costs to be specified are much larger than in the usual setting, but the dimension of the parameters is reduced by considering a convex combination between the costs of the spells and the transition costs, controlled by some w. In the following, the value of w was fixed so as to favor a criterion sensitive to differences in sequencing.

The second set of dissimilarities was focused toward highlighting timing. Four distances were tested: the Hamming distance (**HAM**), based on the number of non-matching states, the Euclidean distance (**EUCLID**), which, in this case, is the squared root of the Hamming, the χ^2 (**CHI2**), which is similar to the two previous dissimilarities except that it gives more weights to the infrequent states, and the Dynamic Hamming distance (**DHD**). For both **EUCLID** and **CHI2** distances, the

sensitivity between duration and timing is controlled through a parameter, L. When $L = 94$ (*i.e.*, in our case the trajectory length), scores are similar to those of the Hamming family regarding timing, when $L = 1$, dissimilarity measures are more sensitive to duration. **DHD**, proposed by [27], generalizes the Hamming distance by considering an OM dissimilarity without insertions or deletions, and with the substitution costs defined at each temporal instant from the corresponding transition matrices. While taking the position in the sequence and time into account, **DHD** has often been criticized for its risk of over-parameterization. In the following, distance parameters used are specified between parentheses.

The third group of dissimilarities was defined to be sensitive to duration of states. Again, four distances were selected: Euclidean and χ^2 distances between state distributions in the whole trajectories, **OMspell** with $\delta = 1$, which is sensitive both to sequencing and duration, and a distance based on the length of the longest common subsequence of two trajectories, **LCS**, as described in [4]. Whereas the Euclidean distance is more sensitive to differences between states with a high duration, the χ^2 gives more importance to rare states.

Figure S1 of Supplementary material illustrates the relations between the different distances on a cosine matrix (computed from the Frobenius dot product) for the school-to-work transition dataset. Within the first group of distances, **SVRspell** with $b = 0$ appears to behave very specifically, and is even very different from the other distances of its own group, except for the **OMspell** with $\delta = 0$. This is explained by the fact that **SVRspell** is sensitive to the sequencing only, and not at all to timing or duration. Furthermore, since it is based on the number of common subsequences only, its principle and computation are quite different from its OM-based counterparts.

The only distance highly correlated to **SVRspell** is the **OMspell** with $\delta = 0$, which is also sensitive to sequencing only. The **OMspell** with $\delta = 0$ is also very different from the distances in its own group, except for **OMspell** with $\delta = 0.1$ and **OMstran**. These two distances introduce some sensitivity with respect to the duration of the spells, while still mainly favoring sequencing. These results are consistent with the conclusions in [52], based on simulated data.

In the second group, the four distances are all very similar, and more particularly **HAM**, **Euclid** and **DHD**. The χ^2-distance stands out because of its particular weighting. We can also note that all four distances in Group 2 are also very similar to **OMstran** and **OMspell** with $\delta = 0.1$, which favor sequencing and duration, and very different from **SVRspell** and **OMspell** with $\delta = 0$, which favor sequencing only.

Within the third group of distances, all distances are also very similar, even if the value of the cosine is a bit lower than within distances in the second group. The distances in the third group are also similar to the ones in the second group, to **OMspell** with $\delta = 0.1$ and to **OMstran** in the first group.

In conclusion, **SVRspell** and **OMspell** with $\delta = 0$ are very different from all other distances, which are globally similar. This indicates that, for this dataset, criteria favoring sequencing are quite opposite to those favoring timing or duration.

6.2 Results of the clusterings

The map obtained with the first four distances is provided in Figure 2. For the sake of conciseness, the other two maps are available in Figures S2 and S3 and in Section S2 of the Supplementary material.

As one may easily see, most of the clusters show smooth sigmoidal transitions between states, and in some cases "sandwich"-like representations. These patterns are inherent to the fact that the first set of distances was built to point out similarities in terms of sequencing and not in terms of timing or duration. A chronogram representation is not a well suited representation in this case because it allows for temporal shifts. Despite that, some clusters of particular interest can be identified, such as the lower-left corner of the map, showing the outcomes of public fixed-term contracts.

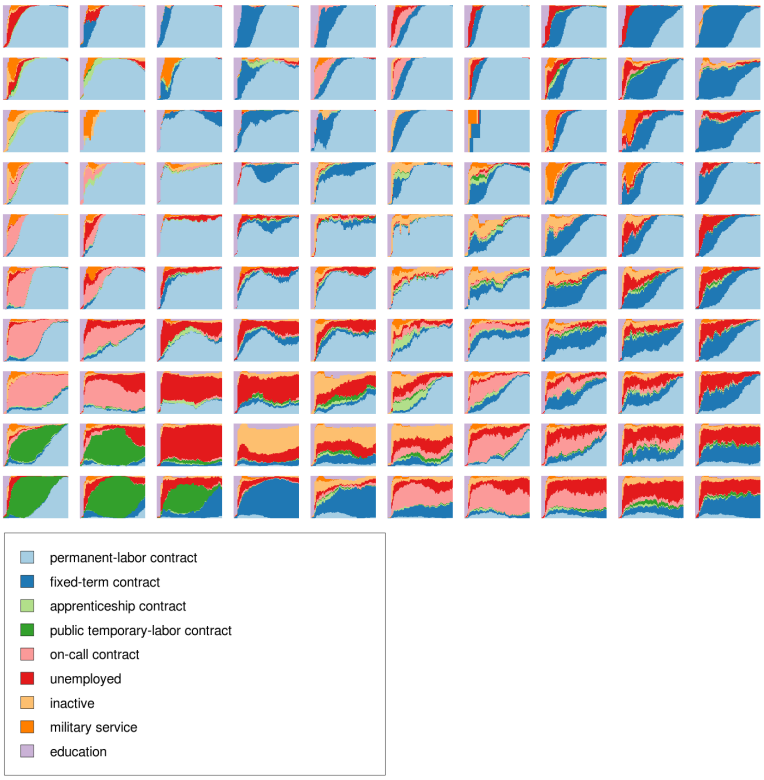

Fig. 2 Final map obtained with the first group of distances.

The final convex combination for the first group of distances was:

$0.27\times$**OMstran**$+0.26\times$**OMspell**$(\delta = \mathbf{0})+0.28\times$**OMspell**$(\delta = \mathbf{0.1})+0.19\times$**SVRspell**.

It gives more weight to the OM-inspired distances, which appear to be more discriminant than the **SVRspell**. As it has already been stressed in the comparison between distances (Figure S1 of the Supplementary material), the **SVRspell** distance behaves very differently from all the other distances and, according to [52], it is very sensitive to sequencing and small random perturbations. The latter may be a reason for which the other dissimilarities appear to be fitter for clustering trajectories based on their sequencing properties.

6.3 Concluding remarks

This chapter has presented a global approach to perform exploratory analysis in the presence of multiple sources of data or of multiple kernels describing different features of the data. The approach has been illustrated on a dataset of categorical time series representing labor-market status of recently graduated people. We have shown the effectiveness of the approach to identify a relevant typography of the dataset, with contrasting results depending on which features the focus is put on. Different dissimilarities led to highlight different characteristics of the trajectories, some less suited to chronogram representations than others. To fully exploit that diversity, alternative representations would be needed, which could be able to better represent similar duration of states or similar global distributions of the trajectories and thus to highlight the distance features.

Acknowledgements. Nathalie Vialaneix would like to sincerely thank Professor Christine Thomas-Agnan for so many interesting discussions and for making her participate to the organization of "Journées de Statistique de la SFdS 2013" (an unforgettable adventure!). From a scientific point of view, Christine's book (written with Alain Berlinet) has been her main starting point for her work on kernel methods and use of spline based regularizations in functional data analysis. For all this, I am more than grateful!

References

1. Abbott, A., Forrest, J.: Optimal matching methods for historical sequences. Journal of Interdisciplinary History **16**, 471–494 (1986)
2. Abbott, A., Tsay, A.: Sequence analysis and optimal matching methods in sociology. Review and Prospect. Sociological Methods and Research **29**(1), 3–33 (2000)
3. Aronszajn, N.: Theory of reproducing kernels. Transactions of the American Mathematical Society **68**(3), 337–404 (1950)
4. Bergroth, L., Hakonen, H., Raita, T.: A survey of longest common subsequence algorithms. In: Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000, pp. 39–48 (2000). DOI 10.1109/SPIRE.2000.878178

5. Berlinet, A., Thomas-Agnan, C.: Reproducing Kernel Hilbert Spaces in Probability and Statistics. Kluwer Academic Publisher, Boston, Norwell, MA, USA / Dordrecht, The Netherlands (2004)
6. Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: 5th annual ACM Workshop on COLT, pp. 144–152. D. Haussler Editor, ACM Press (1992)
7. Boulet, R., Jouve, B., Rossi, F., Villa, N.: Batch kernel SOM and related Laplacian methods for social network analysis. Neurocomputing **71**(7-9), 1257–1273 (2008). DOI doi:10.1016/j.neucom.2007.12.026
8. Chen, J., Bittinger, K., Charlson, E.S., Hoffmann, C., Lewis, J., Wu, G.D., Collman, R.D., Bushman, F.D., Li, H.: Associating microbiome composition with environmental covariates using generalized UniFrac distances. Bioinformatics **28**(16), 2106–2113 (2012). DOI 10.1093/bioinformatics/bts342
9. Chen, Y., Garcia, E., Gupta, M., Rahimi, A., Cazzanti, L.: Similarity-based classification: concepts and algorithm. Journal of Machine Learning Research **10**, 747–776 (2009)
10. Cottrell, M., Letrémy, P.: How to use the Kohonen algorithm to simultaneously analyse individuals in a survey. Neurocomputing **63**, 193–207 (2005)
11. Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall/CRC, Boca Raton, Florida, USA (2001)
12. Cristianini, N., Shawe-Taylor, J.: An Introduction to support vector machines. Cambridge University Press, Cambridge, UK (2000)
13. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means, spectral clustering and normalized cuts. In: W. Kim, R. Kohavi, J. Gehrke, W. DuMouchel (eds.) Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD 2004), pp. 551–556. ACM, New York, NY, USA, Seattle, WA, USA (2004). DOI 10.1145/1014052.1014118
14. Elzinga, C.H., Studer, M.: Spell sequences, state proximities, and distance metrics. Sociological Methods & Research **44**(1), 3–47 (2015)
15. Gabadinho, A., Ritschard, G., Müller, N., Studer, M.: Analyzing and visualizing state sequences in R with TraMineR. Journal of Statistical Software **40**(4) (2011)
16. Goldfarb, L.: A unified approach to pattern recognition. Pattern Recognition **17**(5), 575–582 (1984). DOI 10.1016/0031-3203(84)90056-6
17. Gönen, M., Alpaydin, E.: Multiple kernel learning algorithms. Journal of Machine Learning Research **12**, 2211–2268 (2011).
18. Gönen, M., Margolin, A.A.: Localized data fusion for kernel k-means clustering with application to cancer biology. In: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (eds.) Proceedings of Advances in Neural Information Processing Systems 27 (NIPS 2014), vol. 27, pp. 1305–1313. Curran Associates, Inc. (2014).
19. Hammer, B., Hasenfuss, A.: Topographic mapping of large dissimilarity data sets. Neural Computation **22**(9), 2229–2284 (2010)
20. Hofmann, D., Gisbrecht, A., Hammer, B.: Efficient approximations of robust soft learning vector quantization for non-vectorial data. Neurocomputing **147**, 96–106 (2015). DOI 10.1016/j.neucom.2013.11.044.
21. Huang, H.C., Chuang, Y.Y., Chen, C.S.: Multiple kernel fuzzy clustering. IEEE Transactions on Fuzzy Systems **20**(1), 120–134 (2012). DOI 10.1109/TFUZZ.2011.2170175
22. Kimeldorf, G.S., Wahba, G.: A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. The Annals of Mathematical Statistics **41**(2), 495–502 (1970). DOI 10.1214/aoms/1177697089
23. Kohonen, T.: Self-Organizing Maps, 3rd Edition, vol. 30. Springer, Berlin, Heidelberg, New York (2001)
24. Kohonen, T., Somervuo, P.: Self-organizing maps of symbol strings. Neurocomputing **21**, 19–30 (1998)
25. Kondor, R., Lafferty, J.: Diffusion kernels on graphs and other discrete structures. In: C. Sammut, A. Hoffmann (eds.) Proceedings of the 19th International Conference on Machine Learning, pp. 315–322. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, Sydney, Australia (2002). DOI 10.1.1.57.7612

26. Lavit, C., Escoufier, Y., Sabatier, R., Traissac, P.: The ACT (STATIS method). Computational Statistics and Data Analysis **18**(1), 97–119 (1994). DOI 10.1016/0167-9473(94)90134-1
27. Lesnard, L.: Setting cost in optimal matching to uncover contemporaneous socio-temporal patterns. Sociological Methods & Research **38**(3), 389–419 (2010)
28. Lin, Y., Liu, T., CS., F.: Multiple kernel learning for dimensionality reduction. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**, 1147–1160 (2010)
29. Lozupone, C., Knight, R.: UniFrac: a new phylogenetic method for comparing microbial communities. Applied and Environmental Microbiology **71**(12), 8228–8235 (2005). DOI 10.1128/AEM.71.12.8228-8235.2005
30. Lozupone, C.A., Hamady, M., Kelley, S.T., Knight, R.: Quantitative and qualitative β eiversity measures lead to different insights into factors that structure microbial communities. Applied and Environmental Microbiology **73**(5), 1576–1585 (2007). DOI 10.1128/AEM.01996-06
31. Mac Donald, D., Fyfe, C.: The kernel self organising map. In: Proceedings of 4th International Conference on knowledge-based Intelligence Engineering Systems and Applied Technologies, pp. 317–320 (2000)
32. Mariette, J., Olteanu, M., Villa-Vialaneix, N.: Efficient interpretable variants of online SOM for large dissimilarity data. Neurocomputing **225**, 31–48 (2017). DOI 10.1016/j.neucom.2016.11.014
33. Mariette, J., Rossi, F., Olteanu, M., Villa-Vialaneix, N.: Accelerating stochastic kernel som. In: M. Verleysen (ed.) XXVth European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2017), pp. 269–274. i6doc, Bruges, Belgium (2017)
34. Mariette, J., Villa-Vialaneix, N.: Unsupervised multiple kernel learning for heterogeneous data integration. Bioinformatics **34**(6), 1009–1015 (2018). DOI 10.1093/bioinformatics/btx682
35. Massoni, S., Olteanu, M., Villa-Vialaneix, N.: Which distance use when extracting typologies in sequence analysis? An application to school to work transitions. In: International Work Conference on Artificial Neural Networks (IWANN 2013). Puerto de la Cruz, Tenerife (2013)
36. Needleman, S., Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology **48**(3), 443–453 (1970)
37. Ong, C.S., Mary, X., Canu, S., Smola, A.J.: Learning with non-positive kernels. In: C. Brodley (ed.) Proceedings of the XXIst International Conference on Machine Learning (ICML 2004), p. 81. ACM, New York, NY, USA, Banff, AB, Canada (2004). DOI 10.1145/1015330.1015443
38. L'Hermier des Plantes, H.: Structuration des tableaux à trois indices de la statistique. Ph.D. thesis, Université de Montpellier (1976). Thèse de troisième cycle
39. Rapaport, F., Zinovyev, A., Dutreix, M., Barillot, E., Vert, J.: Classification of microarray data using gene networks. BMC Bioinformatics **8**, 35 (2007). DOI 10.1186/1471-2105-8-35
40. Rossi, F.: 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), *Advances in Intelligent Systems and Computing*, vol. 295, pp. 3–23. Springer Verlag, Berlin, Heidelberg, Mittweida, Germany (2014). DOI 10.1007/978-3-319-07695-9_1
41. Saunders, G., Gammerman, A., Vovk, V.: Ridge regression learning algorithm in dual variables. In: Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98), pp. 515–521. Madison, Wisconsin, USA (1998)
42. Schleif, F.M., Tino, P.: Indefinite proximity learning: a review. Neural Computation **27**(10), 2039–2096 (2015). DOI 10.1162/neco_a_00770
43. Schölkopf, B., Herbrich, R., Smola, A.: A generalized representer theorem. In: D. Heimbold, B. Williamson (eds.) Proceedings of the 14th Conference on Computational Learning Theory (COLT), *Lecture Notes in Computer Science*, vol. 2111, pp. 416–426. Springer Berlin Heidelberg (2001). DOI 10.1007/3-540-44581-1_27
44. Schölkopf, B., Smola, A., Müller, K.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10**(5), 1299–1319 (1998). DOI 10.1162/089976698300017467
45. Schölkopf, B., Tsuda, K., Vert, J.: Kernel Methods in Computational Biology. MIT Press, London, UK (2004)

46. Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge University Press, Cambridge, UK (2004)
47. Smola, A., Kondor, R.: Kernels and regularization on graphs. In: M. Warmuth, B. Schölkopf (eds.) Proceedings of the Conference on Learning Theory (COLT) and Kernel Workshop, Lecture Notes in Computer Science, pp. 144–158. Springer-Verlag Berlin Heidelberg, Washington, DC, USA (2003). DOI 10.1007/978-3-540-45167-9_12
48. Speicher, N.K., Pfeifer, N.: Integrating different data types by regularized unsupervised multiple kernel learning with application to cancer subtype discovery. Bioinformatics **31**(12), i268–i275 (2015). DOI 10.1093/bioinformatics/btv244
49. Speicher, N.K., Pfeifer, N.: Towards multiple kernel principal component analysis for integrative analysis of tumor samples. Journal of Integrative Bioinformatics **14**(2), 20170019 (2017). DOI 10.1515/jib-2017-0019
50. Steinwart, I.: On the influence of the kernel on the consistency of support vector machines. Journal of Machine Learning Research **2**, 67–93 (2001)
51. Steinwart, I.: Support vector machines are universally consistent. Journal of Complexity **18**, 768–791 (2002)
52. Studer, M., Ritschard, G.: What matters in differences between life trajectories: a comparative review of sequence dissimilarity measures. Journal of the Royal Statistical Society: Series A (Statistics in Society) **179**(2), 481–511 (2016). DOI 10.1111/rssa.12125
53. Ultsch, A., Siemon, H.P.: Kohonen's self organizing feature maps for exploratory data analysis. In: Proceedings of International Neural Network Conference (INNC'90), pp. 305–308. Kluwer Academic Press, Dordrecht, The Netherlands (1990)
54. Vert, J., Kanehisa, M.: Extracting active pathways from gene expression data. Bioinformatics **19**(Suppl. 2), ii238–ii244 (2003). DOI 10.1093/bioinformatics/btg1084
55. Villa-Vialaneix, N.: Stochastic self-organizing map variants with the R package SOMbrero. In: J. Lamirel, M. Cottrell, M. Olteanu (eds.) 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (Proceedings of WSOM 2017). IEEE, Nancy, France (2017). DOI 10.1109/WSOM.2017.8020014
56. Wang, B., Zhu, J., Pierson, E., Ramazzotti, D., Batzoglou, S.: Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. Nature Methods **14**, 414–416 (2017). DOI 10.1038/nmeth.4207
57. Williams, C., Seeger, M.: 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).
58. Yu, S., Tranchevent, L., Liu, X., Glanzel, W., Suykens, J.A., de Moor, B., Moreau, Y.: Optimized data fusion for kernel k-means clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(5), 1031–1039 (2012). DOI 10.1109/TPAMI.2011.255
59. Zhao, B., Kwok, J., Zhang, C.: Multiple kernel clustering. In: C. Apte, H. Park, K. Wang, M. Zaki (eds.) Proceedings of the 2009 SIAM International Conference on Data Mining (SDM), pp. 638–649. SIAM, Philadelphia, PA (2009). DOI 10.1137/1.9781611972795.55
60. Zhuang, J., Wang, J., Hoi, S., Lan, X.: Unsupervised multiple kernel clustering. Journal of Machine Learning Research: Workshop and Conference Proceedings **20**, 129–144 (2011)