How Many Dissimilarity/Kernel Self Organizing Map Variants Do We Need?

Fabrice Rossi

How Many Dissimilarity/Kernel Self Organizing Map Variants Do We Need?

Fabrice Rossi

SAMM (EA 4543), Université Paris 1,
90, rue de Tolbiac, 75634 Paris Cedex 13, France
fabrice.rossi@univ-paris1.fr

Abstract. In numerous applicative contexts, data are too rich and too complex to be represented by numerical vectors. A general approach to extend machine learning and data mining techniques to such data is to really on a dissimilarity or on a kernel that measures how different or similar two objects are.
This approach has been used to define several variants of the Self Organizing Map (SOM). This paper reviews those variants in using a common set of notations in order to outline differences and similarities between them. It discusses the advantages and drawbacks of the variants, as well as the actual relevance of the dissimilarity/kernel SOM for practical applications.

Keywords: Self Organizing Map; Dissimilarity data; Pairwise data; Kernel; Deterministic annealing

1 Introduction

Complex data are frequently too rich and too elaborate to be represented in a simple tabular form where each object is described via a fixed set of attributes/variables with numerical and/or nominal values. This is especially the case for relational data when objects of different categories are interconnected by relations of different types. For instance online retailers have interconnected customers and products databases, in which a customer can buy one or several copies of a product, and can also leave some score and/or review of said products.

Adapting data mining and machine learning methods to complex data is possible, but time consuming and complex, both at the theoretical level (e.g., consistency of the algorithms is generally proved only in the Euclidean case) and on a practical point of view (new implementations are needed). Therefore, it is tempting to build generic methods that use only properties that are shared by all types of data.

Two such generic approaches have been used successfully: the dissimilarity based approach and the kernel based approach [42]. Both are based on fairly generic assumptions: the analyst is given a data set on which either a dissimilarity or a kernel is defined. A dissimilarity measures how much two objects differs, while a kernel can be seen as a form a similarity measure, at least in

the correlation sense. Dozens of dissimilarities and kernels have been proposed over the years, covering many types of complex data (see e.g. [15]). Then one needs only to adapt a classical data mining or machine learning method to the dissimilarity/kernel setting in order to obtain a fully generic approach. As a dissimilarity can always be constructed from a kernel, dissimilarity algorithms are probably the more generic ones. A typical example is the k nearest neighbor method which is based only on dissimilarities.

We review in this paper variants of the Self Organizing Map (SOM) that have been proposed following this line of research, that is SOM variants that operate on dissimilarity/kernel data. We discuss whether those variants are really usable and helpful in practice. The paper is organized as follows. Section 2 describes our general setting: dissimilarity data, kernel data and the Self Organizing Map. Section 3 is dedicated to the oldest dissimilarity variant of the SOM, the Median SOM, while Section 4 focuses on the modern variant, the relational SOM. Section 5 presents a different approach to SOM extensions based on the deterministic annealing principle. Section 6 describes kernel based variants of the SOM. An unifying view is provided in Section 7 which shows that the differences between the SOM variants are mainly explained by the optimization strategy rather than by the data properties. Finally Section 8 gathers our personal remarks and insights on the dissimilarity/kernel SOM variants.

2 General setting

The data set under study comprises N data points $x_1, \ldots, x_N$ from an abstract space $\mathcal{X}$. We specify below the two options, namely dissimilarity data and kernel data. We also recall the classical SOM algorithms.

2.1 Dissimilarity data

In the dissimilarity data setting (a.k.a. the pairwise data setting), it is assumed that the data are described indirectly by a square $N \times N$ symmetric matrix D that contains dissimilarities between the data points. The convention is that $D_{ij} = d(x_i, x_j)$, a non negative real number, is high when x_i and x_j are different and low when they are similar. Minimal assumptions on D are symmetry and non negativity of each element. It is also natural to assume some basic ordering, that is that $D_{ii} \leq D_{ij}$ for all i and j, but this is not used in SOM variants. Some theoretical results also need $D_{ii} = 0$ (e.g. [20]), but again this is not a very strong constraint. Notice that one can be given either the dissimilarity function d from $\mathcal{X}^2$ to $\mathbb{R}^+$ or directly the matrix D.

2.2 Kernel data

In the kernel data setting, one is given a *kernel* function k from $\mathcal{X}^2$ to $\mathbb{R}$ which satisfies the following properties:

1. k is symmetric: for all x and y in $\mathcal{X}$, $k(x, y) = k(y, x)$;

2. k is positive definite: for all $m > 0$, all observation set $(x_1, \ldots, x_m) \in \mathcal{X}^m$ and all coefficient set $(\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^m$, $\sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j k(x_i, x_j) \geq 0$.

The most important aspect of the kernel setting lays in the Moore-Aronszajn theorem [3]. It states that a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ can be associated to $\mathcal{X}$ and k through a mapping function ϕ from $\mathcal{X}$ to $\mathcal{H}$ such that $\langle \phi(x), \phi(y) \rangle_{\mathcal{H}} = k(x, y)$ for all x and y in $\mathcal{X}$. The mapping ϕ is called the *feature map*. It enables one to leverage the Hilbert structure of $\mathcal{H}$ in order to build machine learning algorithms on $\mathcal{X}$ indirectly.This can be done in general without using ϕ but rather by relying on k only: this is known as the *kernel trick* (see e.g. [42]).

Notice that the kernel can be used to define a dissimilarity on $\mathcal{X}$ by transporting the Hilbert distance from $\mathcal{H}$. Indeed, it is natural to define d_k on $\mathcal{X}$ by

$$d_k(x, y) = \langle \phi(x) - \phi(y), \phi(x) - \phi(y) \rangle_{\mathcal{H}}. \tag{1}$$

Elementary algebraic manipulations show that

$$d_k(x, y) = k(x, x) + k(y, y) - 2k(x, y), \tag{2}$$

which is an example of the use of the kernel trick to avoid using explicitly ϕ.

The construction of d_k shows that the dissimilarity setting is more general than the kernel setting. It is always possible to use a kernel as the basis of a dissimilarity: all the dissimilarity variants of the SOM can used on kernel data. Therefore, we will focus mainly on dissimilarity algorithms, and then discuss how they relate to their kernel counterparts.

Notice finally that as in the case of the dissimilarity setting, the kernel can be given as a function from $\mathcal{X}$ to $\mathbb{R}$ or as a kernel matrix $K = (K_{ij}) = (k(x_i, x_j))$. In the latter case, K is symmetric and positive definite and is associated to a dissimilarity matrix D_K via equation (2).

2.3 SOM

To contrast its classical setting with the dissimilarity and kernel ones, and to introduce our notations, we briefly recall the SOM principle and algorithm [28]. A SOM is a low dimensional clustered representation of a data set.

One needs first to specify a low dimensional prior structure, in general a regular lattice of K units/neurons positioned in $\mathbb{R}^2$, the $(r_k)_{1 \leq k \leq K}$. The structure induces a time dependent neighborhood function $h_{kl}(t)$ which measures how much the prototype/model associated to unit r_k should be close to the one associated to unit r_l, at step t of the learning algorithm (from 0 for unrelated models to 1 for maximally related ones). We will not discuss here the numerous possible variants for this neighborhood function [28]: if the lattice is made of points r_k in $\mathbb{R}^2$ a classical choice is

$$h_{kl}(t) = exp\left(-\frac{\|r_k - r_l\|^2}{2\sigma^2(t)}\right),$$

where σ increases over time to reduce gradually the influences of the neighbors during learning.

The SOM attaches to each unit/neuron r_k in the prior structure a prototype/model in the data space m_k. The objective of the SOM algorithm is to adapt the values of the models in such a way that each data point is as close as possible to its closest model in the data space (at standard goal in prototype based clustering). In addition if the closest model for the data point x is m_k, then m_l should also be close to x if r_k and r_l are close in the prior structure. In other words, proximities in the prior structure should reflect proximities in the data space and vice versa. The unit/neuron associated to the closest model of a data point is called the *best matching unit* (BMU) for this point. The set of points for which r_k is the BMU defines a cluster in the data space, denoted C_k.

This is essentially achieved via two major algorithms (and dozens of variants). Let us assume that the data space is a classical normed vector space. Then both algorithms initialize the prototypes $(m_k)_{1\leq k\leq K}$ in an "appropriate way" and proceed then iteratively. We will not discuss initialization strategies in this paper.

In the stochastic/online SOM (SSOM), a data point x is selected randomly[1] at each iteration t. Then $c \in \{1,\dots,K\}$ is determined as the index of the best matching unit, that is

$$c = \arg\min_{k\in\{1,\dots,K\}} \|x - m_k(t)\|^2, \tag{3}$$

and all prototypes are updated via

$$m_k(t+1) = m_k(t) + \epsilon(t)h_{kc}(t)(x - m_k(t)), \tag{4}$$

where $\epsilon(t)$ is a learning rate.

In the batch SOM (BSOM), each iteration is made of two steps. In the first step, the best matching unit for each data point x_i is determined as:

$$c_i(t) = \arg\min_{k\in\{1,\dots,K\}} \|x_i - m_k(t)\|^2. \tag{5}$$

Then all prototypes are updated via a weighted average

$$m_k(t+1) = \frac{\sum_{i=1}^N h_{kc_i(t)}(t)x_i}{\sum_{i=1}^N h_{kc_i(t)}(t)}. \tag{6}$$

Obviously, neither algorithm can be applied *as is* on non vector data.

[1] or data points are looped through.

3 The Median SOM

3.1 General principle

It is well known (and obvious) that the prototype update step of the Batch SOM can be considered as solving an optimization problem, namely

$$\forall\ k \in \{1, \ldots, K\},\ m_k(t+1) = \arg\min_{s} \sum_{i=1}^{N} h_{kc_i(t)}(t) \|s - x_i\|^2. \tag{7}$$

This turns the vector space operations involved in equation (6) into an optimization problem that uses only the squared Euclidean norm between prototypes and observations. In an arbitrary space $\mathcal{X}$ with a dissimilarity, $\|s_k - x_i\|^2$ can be replaced by the dissimilarity between s_k and x_i which turns problem (7) into

$$\forall\ k \in \{1, \ldots, K\},\ m_k(t+1) = \arg\min_{s \in \mathcal{X}} \sum_{i=1}^{N} h_{kc_i(t)}(t) d(s, x_i), \tag{8}$$

which is a typical generalized median problem.

However, the most general dissimilarity setting only assumes the availability of dissimilarities between *observations* not between arbitrary points in $\mathcal{X}$. In fact, generating new points in $\mathcal{X}$ might be difficult for complex data such as texts. Then the most general solution consists in looking for the optimal prototypes into the data set rather than in $\mathcal{X}$. The Median SOM [27,29,30] and its variants [12,13] are based on this principle. The Median SOM consists in iterating two steps. In the first step, the best matching unit for each data point x_i is determined as

$$c_i(t) = \arg \min_{k \in \{1, \ldots, K\}} d(x_i, m_k(t)). \tag{9}$$

Then all prototypes are updated by solving the generalized median problem

$$\forall\ k \in \{1, \ldots, K\},\ m_k(t+1) = \arg\min_{x_j} \sum_{i=1}^{N} h_{kc_i(t)}(t) D_{ij}. \tag{10}$$

Notice that each prototype is a data point which means that in equation (9) $d(x_i, m_k(t))$ is in fact a D_{il} for some l.

A variant of the Median SOM was proposed in [1]: rather than solving problem (10), it associates to each unit the generalized median of the corresponding cluster (in other words, it does not take into account the neighborhood structure at this point). Then the BMU of a data point is chosen randomly using the neighborhood structure and the dissimilarities. This means that a data point can be moved from its natural BMU to a nearby one. As far as we know, this variant has not been studied in details.

3.2 Limitations of the Median SOM

The Median SOM has numerous problems. As a batch SOM it is expected to request more iterations to converge than a potential stochastic version (which is not possible in the present context, unfortunately). In addition, it will also exhibit sensitivity to its initial configuration.

There are also problems more specific to the Median SOM. Each iteration of the algorithm has a rather high computational cost: a naive implementation leads to a cost of $O(N^2K + NK^2)$ per iteration, while a more careful one still costs $O(N^2 + NK^2)$ [10]. Numerous tricks can be used to reduce the actual cost per iteration [7,8] but the N^2 factor cannot be avoided without introducing approximations.

Arguably the two main drawbacks of the Median SOM are of a more intrinsic nature. Firstly, restricting the prototypes to be chosen in the data set has some very adverse effects. A basic yet important problem comes from collisions in prototypes [36]: two different units can have the same optimal solution according to equation (10). This corresponds to massive folding of the two dimensional representation associated to the SOM and thus to a sub-optimal data summary. In addition, equation (9) needs a tie breaking rule which will in general increase the cost of BMU determination (see [30] for an example of such a rule). The solution proposed in [36] can be used to avoid those problems at a reasonable computational cost.

A more subtle consequence of the restriction of prototypes to data points is that no unit can remain empty, apart from collided prototypes. Indeed, the BMU of a data point that is used as a prototype should be the unit of which it is the prototype. This means that no interpolation effect can take place in the Median SOM [43,44] a fact that limits strongly the usefulness of visual representations such as the U-matrix [45,46]. For some specific data types such as strings, this can be avoided by introducing ways of generating new data points by some form of interpolations. This was studied in [43,44] together with a stochastic/online algorithm.

A generic solution to lift the prototype restriction is provided by the relational SOM described in Section 4.

3.3 Non metric dissimilarities

The second intrinsic problem of the Median SOM is its reliance on a prototype based representation of a cluster in the dissimilarity context, while this is only justified in the Euclidean context. Indeed let us consider that the N data points $(x_i)_{1\leq i\leq N}$ belong to a Euclidean space. Then for any vector of positive weights $(\beta_i)_{1\leq i\leq N}$, the well known König-Huygens identity states:

$$\sum_{i=1}^{N} \beta_i \left\| \frac{\sum_{j=1}^{N} \beta_j x_j}{\sum_{j=1}^{N} \beta_j} - x_i \right\|^2 = \frac{1}{2} \frac{1}{\sum_{i=1}^{N} \beta_i} \sum_{i=1}^{N} \sum_{j=1}^{N} \beta_i \beta_j \|x_i - x_j\|^2. \qquad (11)$$

This means that

$$\min_m \sum_{i=1}^{N} \beta_i \left\|m - x_i\right\|^2 = \frac{1}{2} \frac{1}{\sum_{i=1}^{N} \beta_i} \sum_{i=1}^{N} \sum_{j=1}^{N} \beta_i \beta_j \|x_i - x_j\|^2. \tag{12}$$

Applied to the SOM, this means that solving[2]

$$(m(t), c(t)) = \arg\min_{m,c} \sum_{k=1}^{K} \sum_{i=1}^{N} h_{kc_i}(t) \|m_k - x_i\|^2, \tag{13}$$

where $m(t) = (m_1(t), \ldots, m_K(t))$ denotes the prototypes and $c = (c_1, \ldots, c_n)$ denotes the BMU mapping, is equivalent to solving

$$c(t) = \arg\min_c \frac{1}{2} \sum_{k=1}^{K} \frac{1}{\sum_{i=1}^{N} h_{kc_i}(t)} \sum_{i=1}^{N} \sum_{j=1}^{N} h_{kc_i}(t) h_{kc_j}(t) \|x_i - x_j\|^2. \tag{14}$$

This second problem makes clear that the classical SOM is not only based on *quantization* but is also optimizing the within pairwise distances in the clusters defined by the BMU mapping. Here h_{kc_i} is considered as a form of membership value of x_i to cluster k, which give the "size" $\sum_{i=1}^{N} h_{kc_i}$ to the cluster k. Then the sum of pairwise distances in each cluster measures the compactness of the cluster in terms of within variance. As the SOM minimizes the sum of those quantities, it can be seen as a *clustering* algorithm[3].

However, the König-Huygens identity does not apply to arbitrary dissimilarities. In other words, the natural dissimilarity version of problem (14) that is

$$c(t) = \arg\min_c \frac{1}{2} \sum_{k=1}^{K} \frac{1}{\sum_{i=1}^{N} h_{kc_i}(t)} \sum_{i=1}^{N} \sum_{j=1}^{N} h_{kc_i}(t) h_{kc_j}(t) d(x_i, x_j), \tag{15}$$

is not equivalent to the Median SOM problem given by

$$(m(t), c(t)) = \arg \min_{m \in \{x_1, \ldots, x_N\}^K, c} \sum_{k=1}^{K} \sum_{i=1}^{N} h_{kc_i}(t) d(x_i, m_k). \tag{16}$$

When the dissimilarity satisfies the triangular inequality this is not a major problem. By virtue of the triangular inequality, we have for all m

$$d(x_i, x_j) \leq d(x_i, m) + d(m, x_j), \tag{17}$$

and therefore for all m

$$\sum_{i=1}^{N} \sum_{j=1}^{N} h_{kc_i}(t) h_{kc_j}(t) d(x_i, x_j) \leq 2 \left(\sum_{i=1}^{N} h_{kc_i}(t) \right) \sum_{j=1}^{N} h_{kc_j}(t) d(x_j, m), \tag{18}$$

[2] The quantity optimized in equation (13) is the energy defined in [25].

[3] This classical analysis mimics the one used to see the k-means algorithm both as a clustering algorithm and as a quantization algorithm.

which shows that

$$\frac{1}{2\sum_{i=1}^{N} h_{kc_i}(t)} \sum_{i=1}^{N}\sum_{j=1}^{N} h_{kc_i}(t)h_{kc_j}(t)d(x_i,x_j) \leq \min_m \sum_{j=1}^{N} h_{kc_j}(t)d(x_j,m). \quad (19)$$

Then the Median SOM is optimizing an upper bound of the cluster oriented quality criterion for dissimilarities. In practice, this means that a good quantization will give compact clusters.

However, when the dissimilarity does not satisfy the triangular inequality, the two criteria are not directly related any more. In fact, one prototype can be close to a set of data points while those points remain far apart from each other. Then doing of form of *quantization* by solving problem (16) is not the same thing as doing a form of *clustering* by solving problem (15). By choosing the prototype based solution, the Median SOM appears to be a quantization method rather than a clustering one. If the goal is to display *prototypes* in an organized way, then this choice make sense (but must be explicit). If the goal is to display *clusters* in an organized way, this choice is intrinsically suboptimal. As pointed out in Section 8, dissimilarity SOMs are not very adapted to prototype display, which puts in question the interest of the Median SOM in particular and of the quantization approach in general.

4 The Relational SOM

The quantification of the prototypes induced by restricting them to data points has quite negative effects described in Section 3.2. The relational approach is a way to address this problem. It is based on the simple following observation [23]. Let the $(x_i)_{1,\dots,N}$ be N points in a Hilbert space equipped with the inner product $\langle .,. \rangle$ and let $y = \sum_{i=1}^{N} \alpha_i x_i$ for arbitrary real valued coefficients $\alpha^T = (\alpha_i)_{1,\dots,N}$ with $\sum_{i=1}^{N} \alpha_i = 1$. Then

$$\langle x_i - y, x_i - y \rangle = (D\alpha)_i - \frac{1}{2}\alpha^T D \alpha, \quad (20)$$

where D is the squared distance matrix given by $D_{ij} = \langle x_i - x_j, x_i - x_j \rangle$. This means that computing the (squared) distance between a linear combination of some data points and any of those data points can be done using only the coefficients of the combination and the (squared) distance matrix between those points.

4.1 Principle

But as shown by equation (6), prototypes in the classical SOM are exactly linear combinations of data points whose coefficients sum to one. It is therefore possible to express the Batch SOM algorithm without using directly the values of the x_i, but rather by keeping the coefficients of the prototypes and using equation (20) and the squared distance matrix to perform all calculations.

Then one can simply apply the so called *relational* version of the algorithm to an arbitrary dissimilarity matrix as if it were a squared euclidean one. This is essentially what is done in [22,23] for the c-means (a fuzzy variant of the k-means) and in [21] for the Batch SOM (and the Batch Neural Gas [11]). Using the concept of pseudo-Euclidean spaces, it was shown in [20] that this general approach can be given a rigorous derivation: it amounts to using the original algorithm (SOM, k-means, etc.) on a pseudo-Euclidean embedding of the data points.

In practice, the Batch relational SOM proceeds by iterating two steps that are very similar to the classical Batch SOM steps. The main difference is that each prototype $m_k(t)$ (at iteration t) is given by a vector of $\mathbb{R}^N$, $\alpha_k(t)$, which represents the coefficients of the linear combination of the x_i in the pseudo-Euclidean embedding. Then the best matching unit computation from equation (5) is replaced by

$$c_i(t) = \arg\min_{k\in\{1,\dots,K\}} \left((D\alpha_k(t))_i - \frac{1}{2}\alpha_k(t)^T D\alpha_k(t) \right), \tag{21}$$

while the prototype update becomes

$$\alpha_k(t+1)_i = \frac{h_{kc_i(t)}}{\sum_{l=1}^{N} h_{kc_l(t)}}. \tag{22}$$

A stochastic/online variant of this algorithms was proposed in [34]. As for the classical SOM, it consists in selecting randomly a data point x_i, computing its BMU c_i (using equation (21)) and updating all prototypes as follows:

$$\alpha_k(t+1)_j = \alpha_k(t)_j + \epsilon(t) h_{kc_i}(t)(\delta_{ij} - \alpha_k(t)_j), \tag{23}$$

where δ_{ij} equals 1 when $i = j$ and 0 in other cases. Notice that is the α_k are initialized so as to sum to one, this is preserved by this update. As shown in [34], the stochastic variant tends to be less sensitive to the initial values of the prototypes. However [34] overlooks that both batch and online relational SOM algorithms share the same computational cost per iteration[4] which negates the traditional computational gain provided by online versions.

4.2 Limitations of the Relational SOM

The Relational SOM solves several problems of the Median SOM. In particular, it is not subject to the quantization effect induced by constraining the prototypes to be data points. As a consequence, it exhibits in practice the same interpolation effects as the classical SOM. The availability of a stochastic version provides also a simple way to reduce the adverse effects of a bad initialization.

However, the relational SOM is very computationally intensive. Indeed, the evaluation of all the $\alpha_k(t)^T D\alpha_k(t)$ costs $O(KN^2)$ operations. Neither the dissimilarity matrix nor the prototype coefficients are sparse and there is no way

[4] the cost reported in [34] for the batch relational SOM is incorrect.

to reduce this costs without introducing approximations. Notice that this cost is per iteration in both the batch and the stochastic versions of the relational SOM. This is K times larger than the Median SOM.

This large cost has motivated research on approximation techniques such as [37]. The most principled approach consists in approximating the calculation of the matrix product via the Nyström technique [50], as explored in [19].

5 Soft Topographic Mapping for Proximity Data

As pointed out in Section 3.3, if an algorithm relies on prototypes with a general possibly non metric dissimilarity, it provides only quantization and not clustering. When organized clusters are looked for, one can try to solve problem (15) directly, that is without relying on prototypes.

5.1 A deterministic annealing scheme

However problem (15) is combinatorial and highly non convex. In particular, the absence of prototypes rules out standard alternating optimization schemes. Following the analysis done in the case of the dissimilarity version of the k-means in [6,26], Graepel et al. introduce in [17,18] a deterministic annealing approach to address problem (15). The approach introduces a mean field approximation which estimates by e_{ik} the effects in the criterion of problem (15) of assigning the data point x_i in cluster k. In addition, it computes soft assignments to the cluster/unit, denoted γ_{ik} for the membership of x_i to cluster k ($\gamma_{ik} \in [0,1]$ and $\sum_{k=1}^{K} \gamma_{ik} = 1$). The optimal mean field is given by

$$e_{ik} = \sum_{s=1}^{K} h_{ks} \sum_{j=1}^{N} b_{js} \left(d(x_i, x_j) - \frac{1}{2} \sum_{l=1}^{N} b_{ls} d(x_j, x_l) \right), \tag{24}$$

where the b_{js} are given by

$$b_{js} = \frac{\sum_{k=1}^{K} \gamma_{jk} h_{ks}}{\sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik} h_{ks}}. \tag{25}$$

Soft assignments are updated according to

$$\gamma_{ik} = \frac{\exp(-\beta e_{ik})}{\sum_{s=1}^{K} \exp(-\beta e_{is})}, \tag{26}$$

where β is an annealing parameter. It plays the role of an inverse temperature and is therefore gradually increased at each step of the algorithm.

In practice, the so-called Soft Topographic Mapping for Proximity Data (STMP) is trained in an iterative batch like procedure. Given an annealing schedule (that is a series of increasing values for β) and initial random values of the mean field, the algorithm iterates evaluating equation (26), then equation (25)

and finally equation (24) for a fixed value of β, until convergence. When this convergence is reached, β is increased and the iterations restart from the current value of the mean field.

Notice in equation (25) that the neighborhood function is *fixed* in this approach, whereas it is evolving with time in most SOM implementations.

5.2 Limitations of the STMP

It is well known that the quality of the results obtained by deterministic annealing are highly dependent on the annealing scheme [35]. It is particularly important to avoid missing transition phases. Graepel et al. have analyzed transition phases in the STMP in [18]. As in [35,26], the first critical temperature is related to a dominant eigenvalue of the dissimilarity matrix. As this is in general a dense matrix, the minimal cost of computing the critical temperature is $O(N^2)$. In addition, each internal iteration of the algorithm is dominated by the update of the mean field according to equation (24). The cost of a full update is in $O(N^2K + NK^2)$. The STMP is therefore computationally intensive. It should be noted however that an approximation of the mean field update that reduces the cost to $O(N^2K)$ is proposed in [18], leading to the same computational cost as the relational SOM.

In addition, as will appear clearly in Section 7.2, the STMP is based on prototypes, even they appear only indirectly. Therefore while it tries to optimize the clustering criterion associated to the SOM, it resorts to a similar quantization quality proxy as the relational SOM.

6 Kernel SOM

As recalled in Section 2.2, the kernel setting is easier to deal with than the dissimilarity one. Indeed the embedding into a Hilbert space $\mathcal{H}$ enables to apply any classical machine learning method to kernel data by leveraging the Euclidean structure of $\mathcal{H}$. The kernel trick allows one to implement those methods efficiently.

6.1 The kernel trick for the SOM

In the case of the SOM, the kernel trick is based on the same fundamental remark that enables the relational SOM (see Section 4.1): in the Batch SOM, the prototypes are linear combinations of the data points. If the initial values of the prototypes are linear combinations of the data points (and not random points), this is also the case for the stochastic/online SOM.

Then assume given a kernel k on $\mathcal{X}$, with its associated Hilbert space $\mathcal{H}$ and mapping ϕ. Implementing the Batch SOM in $\mathcal{H}$ means working on the mapped data set $(\phi(x_i))_{1 \leq i \leq N}$ with prototypes $m_k(t)$ of the form $m_k(t) = \sum_{i=1}^{N} \alpha_{ki}(t)\phi(x_i)$. Then equation (5) becomes

$$c_i(t) = \arg \min_{k \in \{1,\dots,K\}} \|\phi(x_i) - m_k(t)\|_{\mathcal{H}}^2, \tag{27}$$

with

$$\|\phi(x_i) - m_k(t)\|^2_{\mathcal{H}} = k(x_i, x_i) - 2\sum_{j=1}^{N} \alpha_{kj}(t) k(x_k, x_j) + \sum_{j=1}^{N}\sum_{l=1}^{N} \alpha_{kj}(t)\alpha_{kl}(t) k(x_j, x_l). \tag{28}$$

Equation (28) is a typical result of the kernel trick: computing the distance between a data point and a linear combination of the data points can be done using solely the kernel function (or matrix). To our knowledge, the first use of the kernel trick in a SOM context was made in [17].

Notice that equation (6) can also been implemented without using explicitly the mapping ϕ as one needs only the coefficients of the linear combination which are given by

$$\alpha_{ki}(t+1) = \frac{h_{kc_i(t)}}{\sum_{l=1}^{N} h_{kc_l(t)}}, \tag{29}$$

exactly as in equation (22). While the earliest kernel SOM (STMK) in [17] is optimized using deterministic annealing (as the SMTP presented in Section 5), the kernel trick enables the more traditional online SOM [31] and batch SOM [5,32,49] derived from the previous equations.

It should be noted for the sake of completeness that another kernel SOM was proposed in [2]. However, this variant assumes that $\mathcal{X}$ is a vector space and therefore is not applicable to the present setting.

6.2 Limitations of the kernel SOM

As it is built indirectly on a Hilbert space embedding, the kernel SOM does not suffer from constrained prototypes. The stronger assumptions made on kernels compared to dissimilarities guarantee the equivalence between finding good prototypes and finding compact clusters. Kernel SOM has also both online and batch versions.

Then the main limitation of the kernel SOM is its computational cost. Indeed, as for the relational SOM, evaluating the distances in equation (28) has a $O(KN^2)$ cost. The approximation schemes proposed for the relational SOM [19,37] can be used for the kernel SOM at the cost of reduced performances in terms of data representation.

7 Equivalences between SOM variants

It might seem at first that all the variants presented in the previous sections are quite different, both in terms of goals and algorithms. On the contrary, with the exception of the Median SOM which is very specific in some aspects, the variations between the different methods are explained by optimization strategies rather than by hypothesis on the data.

7.1 Relational and kernel methods are equivalent

We have already pointed out that relational SOM and kernel SOM share the very same principle of representing prototypes by a linear combination of the data points. Both cases use the same coefficient update formulas whose structure depends only on the type of the algorithm (batch or online).

The connections are even stronger in the sense that given a kernel, the relational SOM algorithm obtained by using the dissimilarity associated to the kernel is *exactly* identical to the kernel SOM algorithm. Indeed if K is the kernel matrix, then the dissimilarity matrix is given by $D_{ij} = K_{ii} + K_{jj} - 2K_{ij}$. Then for all $\alpha \in \mathbb{R}^N$ such that $\sum_{i=1}^N \alpha_i = 1$ and for all $i \in \{1, \ldots, N\}$

$$\begin{aligned}(D\alpha)_i - \frac{1}{2}\alpha^T D\alpha &= \sum_{j=1}^N D_{ij}\alpha_j - \frac{1}{2}\sum_{j=1}^N\sum_{l=1}^N \alpha_j\alpha_l D_{jl} \\ &= \sum_{j=1}^N (K_{ii} + K_{jj} - 2K_{ij})\alpha_j - \frac{1}{2}\sum_{j=1}^N\sum_{l=1}^N \alpha_j\alpha_l(K_{jj} + K_{ll} - 2K_{jl})\end{aligned}$$

Using $\sum_{i=1}^N \alpha_i = 1$, the first term becomes

$$\sum_{j=1}^N (K_{ii} + K_{jj} - 2K_{ij})\alpha_j = K_{ii} + \sum_{j=1}^N K_{jj}\alpha_j - 2\sum_{j=1}^N K_{ij}\alpha_j.$$

The same condition on α shows that

$$\sum_{j=1}^N\sum_{l=1}^N \alpha_j\alpha_l K_{jj} = \sum_{j=1}^N K_{jj}\alpha_j,$$

and that

$$\sum_{j=1}^N\sum_{l=1}^N \alpha_j\alpha_l K_{ll} = \sum_{l=1}^N K_{ll}\alpha_l.$$

Therefore

$$\sum_{j=1}^N\sum_{l=1}^N \alpha_j\alpha_l(K_{jj} + K_{ll} - 2K_{jl}) = 2\sum_{j=1}^N K_{jj}\alpha_j - 2\sum_{j=1}^N\sum_{l=1}^N \alpha_j\alpha_l K_{jl}.$$

Combining those equations, we end up with

$$(D\alpha)_i - \frac{1}{2}\alpha^T D\alpha = K_{ii} - 2\sum_{j=1}^N K_{ij}\alpha_j + \sum_{j=1}^N\sum_{l=1}^N \alpha_j\alpha_l K_{jl}. \tag{30}$$

The second part of this equation is exactly $\|\phi(x_i) - m\|^2_{\mathcal{H}}$ when $m = \sum_{j=1}^N \alpha_j\phi(x_j)$ as recalled in equation (28). Therefore, the best matching unit determination in

the relational SOM according to equation (21) is exactly equivalent to the BMU determination in the kernel SOM according to equation (27). This shows the equivalence between the two algorithms (in both batch and online variants).

This equivalence shows that the batch relational SOM from [21] is a rediscovery of the batch kernel SOM from [32], while the online relational SOM from [34] is a rediscovery of the online kernel SOM from [31]. Results from [20] show that those rediscoveries are in fact *generalizations* of kernel SOM variants as they extend the Hilbert embedding to the more general pseudo-Euclidean embedding. In practice, there is no reason to distinguish the kernel SOM from the relational SOM.

7.2 STMP is a prototype based approach

On the surface, the STMP described in Section 5 looks very different from relational/kernel approaches as it tries to address the combinatorial optimization problem (15) rather than the different problem (16) associated to the generalized median. However, as analyzed in details in [20], the STMP differs from the relational approach only by the use of deterministic annealing, not by the absence of prototypes.

A careful analysis of equations (24) and (22) clarifies this point. Indeed, let us consider $\alpha_s = (b_{js})_{1\leq j\leq N}^T$ as the coefficient vector for a linear combination of the data points x_j embedded in the pseudo-Euclidean space associated to the dissimilarity matrix D. Then

$$\sum_{j=1}^{N} b_{js}\left(d(x_i,x_j)-\frac{1}{2}\sum_{l=1}^{N} b_{ls}d(x_j,x_l)\right)=(D\alpha_s)_i-\frac{1}{2}\alpha_s^T D\alpha_s.$$

The right hand part is the distance in the pseudo-Euclidean space between the prototype associated to α_s and x_i. Then e_{ik} in equation (24) is a weighted average of distances between x_i and each of the α_s, where the weights are given by the neighborhood function. As pointed out in [20], this can be seen as a relational extension of the assignment rule proposed by Heskes and Kappen in [25].

However, rather than using crisp assignments to a best matching unit with the lowest value of e_{ik}, the STMP uses a soft maximum strategy implemented by equation (26) to obtain assignment probabilities γ_{ik}. Those are used in turn to update the coefficients of the prototypes in equation (25).

In fact the three algorithms proposed in [17] are all based on the same deterministic annealing scheme, with an initial implementation in $\mathbb{R}^p$ (the STVQ) and two generalization in the Hilbert space associated to a kernel (STMK) and in the pseudo-Euclidean space associated to a dissimilarity (STMP). The discussion of the previous section shows that the kernel and the dissimilarity variants are strictly equivalent.

7.3 Summary

We summarizes in the following tables the variants of the SOM discussed in this paper. Table 1 maps a data type and an optimization strategy to a SOM variant.

Relational variants include here the kernel case. Table 2 gives the computational costs of one iteration of the SOM variants.

		Data type		
		$\mathbb{R}^p$ data	Kernel	Dissimilarity
Optimization strategy	Online	online SOM	online relational SOM [31,34]	
	Batch	batch SOM	batch relational SOM [21,32]	
	Batch	NA	NA	Median SOM [27]
	Deterministic annealing	STVQ [17]	STMK [17]	STMP [17]

Table 1. Variants of the SOM

Algorithm	Assignment cost	Prototype update cost
Batch SOM	$O(NKp)$	$O(NKp)$
Online SOM	$O(Kp)$	$O(Kp)$
Median SOM	$O(NK)$	$O(N^2 + NK^2)$
Batch relational SOM	$O(N^2K)$	$O(NK)$
Online relational SOM	$O(N^2K)$	$O(NK)$
STVQ	$O(NKp + NK^2)$	$O(NKp + NK^2)$
STMK/STMP	$O(N^2K + NK^2)$	$O(NK^2)$

Table 2. Computational complexity of SOM variants for N data points, K units and in $\mathbb{R}^p$ for the classical SOM.

8 Discussion

Even if the kernel approaches are special cases of the relational ones, we have numerous candidates for dissimilarity processing with the SOM. We discuss those variants in this section.

8.1 Median SOM

In our opinion, there is almost no reason to use the Median SOM in practice, except possibly the reduced computational burden compared to the relational SOM ($O(N^2)$ compared to $O(N^2K)$ for the dominating terms). Indeed, the Median SOM suffers from constraining the prototypes to be data points and gives in general lower performances than the relational/kernel SOM as compared

to a ground truth or based on the usability of the results (see for instance [19,34,49]). The lack of interpolation capability is particularly damaging as it prevents in general to display gaps between natural clusters with u-matrix like visual representation [45,46].

For large data sets, the factor K increase in the cost of one iteration of the relational SOM compared to the median SOM could be seen as a strong argument for the latter. In our opinion, approximation techniques [19,37] are probably a better choice. This remains however to be tested as to our knowledge the effects of the Nyström approximation have only been studied extensively for the relational neural gas and the relational GTM [16,19,40].

8.2 Optimization strategy

To our knowledge, no systematic study of the influence of the optimization strategy has been conducted for SOM variants, even in the case of numerical data. In this latter case, it is well known that the online/stochastic SOM is less sensitive to initial conditions than the batch SOM. It is also generally faster to converge and leads in general to a better overall topology preservation [14]. Similar results are observed in the dissimilarity case in [34]. It should be noted however that both analyses use only random initializations while it is well known (see e.g. [28]) that a PCA[5] based initialization gives much better results than a random one in the case of the batch SOM. It is also pointed in [28] that the neighborhood annealing schedule as some strong effects on topology preservation in the batch SOM. Therefore, in terms of the final quality of the SOM, it is not completely obvious that an online solution will provide better results than a batch one.

In addition, the relational setting negates the computational advantage of the online SOM versus the batch SOM. Indeed in the numerical case, one epoch of the online SOM (a full presentation of all the data points) has roughly the same cost as one iteration of the batch SOM. As the online SOM converges generally with a very small number of epochs, its complete computational cost is lower than the batch SOM. On the contrary, the cost of the relational SOM is dominated by the calculation of $\alpha^T D\alpha$ in equation (21). In the batch relational SOM this quantity can be computed one time per prototype and per iteration, leading to a cost of $O(N^2K)$ per iteration (this is overlooked in [34] which reports erroneously a complexity of $O(N^3K)$ per iteration). In the online version, it has also to be computed for each data point (because of the prototype update that takes place after each data point presentation). This means that one epoch of the online relational SOM costs N times more than one iteration of the batch relational SOM. We think therefore that a careful implementation of the batch relational SOM should outperform the online version, provided the initialization is conducted properly.

Comparisons of the online/batch variants with the deterministic annealing variants is missing, as far as we know. The extensive simulations conducted in

[5] PCA initialization is easily adapted to the relational case, as it was for kernel data [41].

[20] compare the relational neural gas to the dissimilarity deterministic annealing clustering of [6,26]. Their conclusion is the one expected from similar comparisons done on numerical data [35]: the sophisticated annealing strategy of deterministic annealing techniques leads in general to better solutions provided the critical temperatures are properly identified. This comes with a largely increased cost, not really because of the cost per iterations but rather because the algorithm comprises two loops: an inner loop for a given temperature and an outer annealing loop. Therefore the total number of iterations is in general of an order of magnitude higher than with classical batch algorithms (see also [38] for similar results in the context of a graph specific variant of the SOM principle). It should be also noted that in all deterministic variants proposed in [17], the neighborhood function is not adapted during learning. The effects of this choice on the usability of the final results remain to be studied.

To summarize, our opinion is that one should prefer a careful implementation of the batch relational SOM, paired with a PCA like algorithm for initialization and using the Nyström approximation for large data sets. Further experimental work is needed to validate this choice.

8.3 Clustering versus quantization

As explained in Section 3.3, an algorithm that resorts (directly or indirectly) on prototypes for an arbitrary dissimilarity does in fact of form of *quantization* rather than a form of *clustering*. To our knowledge, no attempt has been made to minimize directly the prototype free criterion used in problem (15) and we can only speculate on this point.

We should first note that in the case of classical clustering, it has been shown in [9] that optimizing directly the criterion from problem (15) in its k-means simplified form gives better results than using the relational version of the k-means. While the computational burden of both approaches are comparable, the direct optimization of the pairwise dissimilarities criterion is based on a much more sophisticated algorithm which combines state-of-the-art hierarchical clustering [33] with multi-level refinement from graph clustering [24].

Assuming such a complex technique could be used to train a SOM like algorithm, one would obtain in the end a set of non empty clusters, organized according to a lattice in 2 dimensions, something similar to what can be obtained with the Median SOM. While the clusters would have a better quality, no interpolation between them would be possible, as in the Median SOM.

8.4 How useful are the results?

In our personal opinion, the main interest of the SOM is to provide rich and yet readable visual representations of complex data [47,48]. Unfortunately, the visualization possibilities are reduced in the case of dissimilarity data.

The main limitation is that for arbitrary data in an abstract space $\mathcal{X}$, one cannot assume that an element of $\mathcal{X}$ can be easily represented visually. Then even the Median SOM prototypes (which are data points) cannot be visualized. As the

prototypes (in all the variants) do not have meaningful coordinates, component planes cannot be used.

In fact, the only aspects of the results that can be displayed as in the case of numerical data are dissimilarities between prototypes (in U matrix like displays [45]) as well as numerical characteristics of the clusters (size, compactness, etc.). But as pointed out in [46], among others, this type of visualization is interesting mainly when the SOM uses a large number of units. While this is possible with the relational SOM, it implies a high computational because of the dominating $O(N^2K)$ term. The case of deterministic annealing versions of the SOM is even more problematic with the $O(NK^2)$ complexity term induced by the soft memberships.

In some situations, specific data visualization techniques can be built upon the SOM's results. For instance by clustering graph nodes via a kernel/dissimilarity SOM, one can draw a clustered graph representation, as was proposed in [5]. However, it has been shown in this case that specialized models derived from the SOM [38] or simpler dual approaches based on graph clustering and graph visualization [39] give in general better final results.

To summarize, our opinion is that the appeal of a generic dissimilarity SOM is somewhat reduced by the limited visualization opportunity it offers, compared to the traditional SOM. Further work is needed to explore whether classical visualization techniques, e.g. brushing and linking [4] could be used to provide more interesting displays based on the dissimilarity SOM.

9 Conclusion

We have reviewed in this paper the main variants of the SOM that are adapted to dissimilarity data and to kernel data. Following [20], we have shown that the variants differ more in terms of their optimisation strategy that in other aspects. We have recalled in particular that kernel variants are strictly identical to their relational counterpart. Taking into account computational aspects and known experimental results, our opinion is that the best solution is the batch relational SOM coupled with a structured initialization (PCA like) and with the Nyström approximation for large data sets and thus that we need one dissimilarity/kernel SOM variant only.

However, as discussed above, the practical usefulness of the dissimilarity SOM is reduced compared to the numerical SOM as most of the rich visual representations associated to the SOM of not available for its dissimilarity version. Without improvement in its visual outputs, it is not completely clear if the dissimilarity SOM serves a real practical purpose beyond its elegant generality and simplicity.

References

1. Ambroise, C., Govaert, G.: Analyzing dissimilarity matrices via Kohonen maps. In: Proceedings of 5th Conference of the International Federation of Classification Societies (IFCS 1996). vol. 2, pp. 96–99. Kobe (Japan) (March 1996)

2. Andras, P.: Kernel-Kohonen networks. International Journal of Neural Systems 12, 117–135 (2002)
3. Aronszajn, N.: Theory of reproducing kernels. Transactions of the American Mathematical Society 68(3), 337–404 (May 1950)
4. Becker, A., Cleveland, S.: Brushing scatterplots. Technometrics 29(2), 127–142 (1987)
5. 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 (March 2008)
6. Buhmann, J.M., Hofmann, T.: A maximum entropy approach to pairwise data clustering. In: Proceedings of the International Conference on Pattern Recognition. vol. II, pp. 207–212. IEEE Computer Society Press, Hebrew University, Jerusalem (Israel) (1994)
7. Conan-Guez, B., Rossi, F.: Speeding up the dissimilarity self-organizing maps by branch and bound. In: Sandoval, F., Prieto, A., Cabestany, J., Graña, M. (eds.) Computational and Ambient Intelligence (Proceedings of 9th International Work-Conference on Artificial Neural Networks, IWANN 2007). Lecture Notes in Computer Science, vol. 4507, pp. 203–210. Springer Berlin / Heidelberg, San Sebastián (Spain) (6 2007)
8. Conan-Guez, B., Rossi, F.: Accélération des cartes auto-organisatrices sur tableau de dissimilarités par séparation et évaluation. Revue des Nouvelles Technologies de l'Information pp. 1–16 (6 2008), RNTI-C-2 Classification : points de vue croisés. Rédacteurs invités : Mohamed Nadif et François-Xavier Jollois
9. Conan-Guez, B., Rossi, F.: Dissimilarity clustering by hierarchical multi-level refinement. In: Proceedings of the XXth European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2012). pp. 483–488. Bruges, Belgique (4 2012)
10. Conan-Guez, B., Rossi, F., El Golli, A.: Fast algorithm and implementation of dissimilarity self-organizing maps. Neural Networks 19(6–7), 855–863 (July–August 2006)
11. Cottrell, M., Hammer, B., Hasenfuß, A., Villmann, T.: Batch and median neural gas. Neural Networks 19(6), 762–771 (2006)
12. El Golli, A., Conan-Guez, B., Rossi, F.: Self organizing map and symbolic data. Journal of Symbolic Data Analysis 2(1) (November 2004)
13. El Golli, A., Conan-Guez, B., Rossi, F.: A self organizing map for dissimilarity data. In: Banks, D., House, L., McMorris, F.R., Arabie, P., Gaul, W. (eds.) Classification, Clustering, and Data Mining Applications (Proceedings of IFCS 2004). pp. 61–68. IFCS, Springer, Chicago, Illinois (USA) (July 2004)
14. Fort, J.C., Letremy, P., Cottrell, M.: Advantages and drawbacks of the batch kohonen algorithm. In: Proceedings of Xth European Symposium on Artificial Neural Networks (ESANN 2002). vol. 2, pp. 223–230 (2002)
15. Gärtner, T., Lloyd, J.W., Flach, P.A.: Kernels and distances for structured data. Machine Learning 57(3), 205–232 (2004)
16. Gisbrecht, A., Mokbel, B., Schleif, F.M., Zhu, X., Hammer, B.: Linear time relational prototype based learning. Int. J. Neural Syst. 22(5) (2012)
17. Graepel, T., Burger, M., Obermayer, K.: Self-organizing maps: Generalizations and new optimization techniques. Neurocomputing 21, 173–190 (November 1998)
18. Graepel, T., Obermayer, K.: A stochastic self-organizing map for proximity data. Neural Computation 11(1), 139–155 (1999)

19. Hammer, B., Gisbrecht, A., Hasenfuss, A., Mokbel, B., Schleif, F.M., Zhu, X.: Topographic mapping of dissimilarity data. In: Laaksonen, J., Honkela, T. (eds.) Advances in Self-Organizing Maps, Lecture Notes in Computer Science, vol. 6731, pp. 1–15. Springer Berlin Heidelberg (2011)
20. Hammer, B., Hasenfuss, A.: Topographic mapping of large dissimilarity data sets. Neural Computation 22(9), 2229–2284 (2010)
21. Hammer, B., Hasenfuss, A., Rossi, F., Strickert, M.: Topographic processing of relational data. In: Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 07). Bielefeld (Germany) (9 2007)
22. Hathaway, R.J., Bezdek, J.C.: Nerf c-means: Non-euclidean relational fuzzy clustering. Pattern Recognition 27(3), 429–437 (March 1994)
23. Hathaway, R.J., Davenport, J.W., Bezdek, J.C.: Relational duals of the c-means clustering algorithms. Pattern Recognition 22(2), 205–212 (1989)
24. Hendrickson, B., Leland, R.: A multilevel algorithm for partitioning graphs. In: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM). Supercomputing '95, ACM, New York, NY, USA (1995), http://doi.acm.org/10.1145/224170.224228
25. Heskes, T., Kappen, B.: Error potentials for self-organization. In: Proceedings of 1993 IEEE International Conference on Neural Networks (Joint FUZZ-IEEE'93 and ICNN'93 [IJCNN93]). vol. III, pp. 1219–1223. IEEE/INNS, San Francisco, California (1993)
26. Hofmann, T., Buhmann, J.M.: Pairwise data clustering by deterministic annealing. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(1), 1–14 (January 1997)
27. Kohonen, T.: Self-organizing maps of symbol strings. Technical report A42, Laboratory of computer and information science, Helsinki University of technology, Finland (1996)
28. Kohonen, T.: Self-Organizing Maps, Springer Series in Information Sciences, vol. 30. Springer, third edn. (2001)
29. Kohonen, T., Somervuo, P.J.: Self-organizing maps of symbol strings. Neurocomputing 21, 19–30 (1998)
30. Kohonen, T., Somervuo, P.J.: How to make large self-organizing maps for nonvectorial data. Neural Networks 15(8), 945–952 (2002)
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. Martín-Merino, M., Muñoz, A.: Extending the som algorithm to non-euclidean distances via the kernel trick. In: Pal, N., Kasabov, N., Mudi, R., Pal, S., Parui, S. (eds.) Neural Information Processing, Lecture Notes in Computer Science, vol. 3316, pp. 150–157. Springer Berlin Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-30499-9_22
33. Müllner, D.: Modern hierarchical, agglomerative clustering algorithms. Lecture Notes in Computer Science 3918(1973), 29 (2011), http://arxiv.org/abs/1109.2378
34. Olteanu, M., Villa-Vialaneix, N., Cottrell, M.: On-line relational som for dissimilarity data. In: Estévez, P.A., Príncipe, J.C., Zegers, P. (eds.) Advances in Self-Organizing Maps, Advances in Intelligent Systems and Computing, vol. 198, pp. 13–22. Springer Berlin Heidelberg (2013)
35. Rose, K.: Deterministic annealing for clustering, compression,classification, regression, and related optimization problems. Proceedings of the IEEE 86(11), 2210–2239 (November 1998)

36. Rossi, F.: Model collisions in the dissimilarity SOM. In: Proceedings of XVth European Symposium on Artificial Neural Networks (ESANN 2007). pp. 25–30. Bruges (Belgium) (4 2007)
37. Rossi, F., Hasenfuss, A., Hammer, B.: Accelerating relational clustering algorithms with sparse prototype representation. In: Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 07). Bielefeld (Germany) (9 2007)
38. Rossi, F., Villa-Vialaneix, N.: Optimizing an organized modularity measure for topographic graph clustering: a deterministic annealing approach. Neurocomputing 73(7–9), 1142–1163 (3 2010)
39. Rossi, F., Villa-Vialaneix, N.: Représentation d'un grand réseau à partir d'une classification hiérarchique de ses sommets. Journal de la Société Française de Statistique 152(3), 34–65 (12 2011)
40. Schleif, F.M., Gisbrecht, A.: Data analysis of (non-)metric proximities at linear costs. pp. 59–74. Proceedings of SIMBAD 2013, Springer (2013)
41. Schölkopf, B., Smola, A., Müller, K.R.: Kernel principal component analysis. In: Artificial Neural Networks—ICANN'97, pp. 583–588. Springer (1997)
42. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
43. Somervuo, P.J.: Self-organizing map of symbol strings with smooth symbol averaging. In: Workshop on Self-Organizing Maps (WSOM'03). Hibikino, Kitakyushu, Japan (September 2003)
44. Somervuo, P.J.: Online algorithm for the self-organizing map of symbol strings. Neural Networks 17(1231–1239) (2004)
45. 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 (1990)
46. Ultsch, A., Mörchen, F.: Esom-maps: tools for clustering, visualization, and classification with emergent som. Tech. Rep. 46, Department of Mathematics and Computer Science, University of Marburg, Germarny (2005)
47. Vesanto, J.: Som-based data visualization methods. Intelligent Data Analysis 3(2), 111–126 (1999)
48. Vesanto, J.: Data Exploration Process Based on the Self–Organizing Map. Ph.D. thesis, Helsinki University of Technology, Espoo (Finland) (May 2002), acta Polytechnica Scandinavica, Mathematics and Computing Series No. 115
49. Villa, N., Rossi, F.: A comparison between dissimilarity som and kernel som for clustering the vertices of a graph. In: Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 07). Bielefeld (Germany) (9 2007)
50. Williams, C., Seeger, M.: Using the nyström method to speed up kernel machines. In: Advances in Neural Information Processing Systems 13 (2001)