A comparison between dissimilarity SOM and kernel SOM for clustering the vertices of a graph

Nathalie Villa[(1)] and Fabrice Rossi[(2)]
[(1)] Institut de Mathématiques de Toulouse, Université Toulouse III
118 route de Narbonne, 31062 Toulouse cedex 9, France
[(2)] Projet AxIS, INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 Le Chesnay cedex, France
email: [(1)]nathalie.villa@math.ups-tlse.fr, [(2)]fabrice.rossi@inria.fr

Keywords: kernel SOM, dissimilarity, graph

***Abstract*—** Flexible and efficient variants of the Self Organizing Map algorithm have been proposed for non vector data, including, for example, the dissimilarity SOM (also called the Median SOM) and several kernelized versions of SOM. Although the first one is a generalization of the batch version of the SOM algorithm to data described by a dissimilarity measure, the various versions of the second ones are stochastic SOM. We propose here to introduce a batch version of the kernel SOM and to show how this one is related to the dissimilarity SOM. Finally, an application to the classification of the vertices of a graph is proposed and the algorithms are tested and compared on a simulated data set.

1 Introduction

Despite all its qualities, the original Self Organizing Map (SOM, [13]) is restricted to vector data and cannot therefore be applied to dissimilarity data for which only pairwise dissimilarity measures are known, a much more general setting that the vector one. This motivated the introduction of modified version of the batch SOM adapted to such data. Two closely related *dissimilarity SOM* were proposed in 1996 [12, 1], both based on the generalization of the definition of the mean or median to any dissimilarity measure (hence the alternative name *Median SOM*). Further variations and improvements of this model are proposed in [11, 6, 8].

Another way to build a SOM on non vector data is to use the kernelized version of the algorithm. Kernel methods have become very popular in the past few years and numerous learning algorithm (especially supervised ones) have been "kernelized": original data are mapped into an high dimensional feature space by the way of a nonlinear feature map. Both the high dimensional space and the feature map are obtained implicitly from a kernel function. The idea is that difficult problems can become linear ones while being mapped nonlinearly into high dimensional spaces. Classical (often linear) algorithms are then applied in the feature spaces and the chosen kernel is used to compute usual operations such as dot products or norms; this kernelization provides extensions of usual linear statistical tools into nonlinear ones. This is the case, among others, for Support Vector Machine (SVM, [20]) which corresponds to linear discrimination or Kernel PCA ([17]) which is built on Principal Component Analysis. More recently, kernelized version of the SOM has been studied: [10] first proposes a kernelized version of SOM that aims at optimizing the topographic mapping. Then, [2, 16] present kernel SOM that applies to the images of the original data by a mapping function; they obtain improvements in classification performances of the algorithm. [15, 23] also studied these algorithms: the first one gives a comparison of various kernel SOM on several data sets for classification purposes and the second proves the equivalence between kernel SOM and self-organizing mixture density network.

In this work, we present a batch kernel SOM algorithm and show how this algorithm can be seen as a particular version of the dissimilarity SOM (section 2). We target specifically non vector data, more precisely vertices of a graph for which kernels can be used to define global proximities based on the graph structure itself (section 3.1). Kernel SOM provides in this context an unsupervised classification algorithm that is able to cluster the vertices of a graph into homogeneous proximity groups. This application is of a great interest as graphs arise naturally in many settings, especially in studies of social networks such as World Wide Web, scientific network, P2P network ([3]) or medieval peasant community ([4, 22]). We finally propose to explore and compare the efficiency of these algorithms to this kind of problems on a simulated example (section 3.2).

2 A link between kernel SOM and Dissimilarity SOM

In the following, we consider n input data, $x_1, \ldots, x_n$ from an arbitrary input space, $\mathcal{G}$. In this section, we present self-organizing algorithms using *kernels*, i.e. functions $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ such that are symmetric ($k(x, x') = k(x', x)$)

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

and positive (for all $m \in \mathbb{N}$, all $x_1, \ldots, x_m \in \mathcal{G}$ and all $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$, $\sum_{i=1}^m \alpha_i \alpha_j k(x_i, x_j) \geq 0$).

These functions are dot products of a mapping function, ϕ, which is often nonlinear. More precisely, it exists an Hilbert space, $(\mathcal{H}, \langle ., . \rangle)$, called a *Reproducing Kernel Hilbert Space* (RKHS), and a mapping function $\phi : \mathcal{G} \to \mathcal{H}$ such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$. Then, algorithms that use the input data by the way of their norms or dot products are easily kernelized using the images by ϕ of the original data set: ϕ and $\mathcal{H}$ are not explicitely known as the operations are defined by the way of the kernel function.

2.1 On-line kernel SOM

A kernel SOM based on the k-means algorithm has been first proposed by [16]. The input data of this algorithm are the images by ϕ of $x_1, \ldots, x_n$ and, as in the original SOM, they are mapped into a low dimensional grid made of M neurons, $\{1, \ldots, M\}$, which are related to each others by a neighborhood relationship, h. Each neuron j is represented by a prototype in the feature space $\mathcal{H}$, p_j, which is a linear combination of $\{\phi(x_1), \ldots, \phi(x_n)\}$: $p_j = \sum_{i=1}^n \gamma_{ji} \phi(x_i)$. This leads to the following algorithm:

Algorithm 1: On-line kernel SOM

(1) For all $j = 1, \ldots, M$ and all $i = 1, \ldots, n$, initialize γ_{ji}^0 randomly in $\mathbb{R}$;

(2) For $l = 1, \ldots, L$, do

(3) **assignment step**: x_l is assigned to the neuron, $f^l(x_l)$ which has the closest prototype:

$$f^l(x_l) = \arg \min_{j=1,\ldots,M} \|\phi(x_l) - p_j^{l-1}\|$$

(4) **representation step**: for all $j = 1, \ldots, M$, the prototype p_j is recomputed: for all $i = 1 \ldots, n$,

$$\gamma_{ji}^l = \gamma_{ji}^{l-1} + \alpha(l) h(f^l(x_l), j) \left(\delta_{il} - \gamma_{ji}^{l-1}\right)$$

End for.

where step (3) leads to the minimization of $\sum_{i,i'=1}^n \gamma_{ji}^{l-1} \gamma_{ji'}^{l-1} k(x_i, x_{i'}) - 2 \sum_{i=1}^n \gamma_{ji}^{l-1} k(x_i, x_l)$. As shown in [23], the kernel SOM can be seen as a result of minimizing the energy $\mathcal{E} = \sum_{j=1}^n h(f(x), j) \|\phi(x) - p_j\|^2$ stochastically.

Another version of the kernel SOM is also proposed by [2]: it uses prototypes chosen in the original set and then computes the algorithm with the images by ϕ of this prototype. It comes from the minimization of the following energy $\mathcal{E} = \sum_{j=1}^n h(f(x), j) \|\phi(x) - \phi(p_j)\|^2$.

2.2 Dissimilarity SOM with dissimilarities based on kernels

Dissimilarity SOM ([11, 6, 8]) is a generalization of the batch version of SOM to data described by a dissimilarity measure. We assume given, for all $i, i' = 1, \ldots, n$, a measure $\delta(x_i, x_{i'})$, that is symmetric ($\delta(x, x') = \delta(x', x)$), positive ($\delta(x, x') \geq 0$) and such that $\delta(x, x) = 0$. The Dissimilarity SOM proceeds as follows:

Algorithm 2: Dissimilarity SOM

(1) For all $j = 1, \ldots, M$, initialize p_j^0 randomly to one of the element of the data set $\{x_1, \ldots, x_n\}$;

(2) For $l = 1, \ldots, L$, do

(3) **assignment step**: for all $i = 1, \ldots, n$, x_i is assigned to the neuron, $f^l(x_i)$ which has the closest prototype:

$$f^l(x_i) = \arg \min_{j=1,\ldots,M} \delta(x_i, p_j^{l-1})$$

(4) **representation step**: for all $j = 1, \ldots, M$, the prototype p_j is recomputed:

$$p_j^l = \arg \min_{x \in (x_{i'})_{i'=1,\ldots,n}} \sum_{i=1}^n h(f^l(x_i), j) \delta(x_i, x)$$

End for.

As shown in step (4), the purpose is to choose prototypes in the data set that minimize the generalized energy $\mathcal{E} = \sum_{j=1}^M \sum_{i=1}^n h(f(x_i), j) \delta(x_i, p_j)$.

[6, 8] propose variants of this algorithm: the first one allows the use of several prototypes for a single neuron and the second describes a faster version of the algorithm.

In [22], a dissimilarity based on a kernel is described: it is designed for the clustering of the vertices of a graph. To construct their dissimilarity, the authors take advantage of the fact that the kernel can be interpreted as a norm; then computing the distance induced by this norm leads to the definition of a dissimilarity measure on $\{x_1, \ldots, x_n\}$:

$$\begin{aligned} \delta_{med}(x, x') &= \|\phi(x) - \phi(x')\| \\ &= \sqrt{k(x, x) + k(x', x') - 2k(x, x')}. \end{aligned} \quad (1)$$

We can also define a variant of this dissimilarity measure by, for all x, x' in $\mathcal{G}$,

$$\begin{aligned} \delta_{mean}(x, x') &= \|\phi(x) - \phi(x')\|^2 \\ &= k(x, x) + k(x', x') - 2k(x, x'). \end{aligned} \quad (2)$$

We now show that the dissimilarity SOM based on this last measure can be seen as a particular case of a batch version of the kernel SOM.

2.3 Batch kernel SOM

Replacing the value of the dissimilarity (2) in the representation step of the Dissimilarity SOM algorithm leads to the following equation:

$$p_j^l = \arg \min_{x \in (x_{i'})_{i'=1,\ldots,n}} \sum_{i=1}^n h(f^l(x_i), j) \|\phi(x_i) - \phi(x)\|^2.$$

In this equation, the prototypes are the images by ϕ of some vertices; if we now allow the prototypes to be

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

linear combinations of $\{\phi(x_i)\}_{i=1,\dots,n}$ as in the kernel SOM (section 2.1), the previous equation becomes $p_j^l = \sum_{i'=1}^n \gamma_{ji'}^l \phi(x_{i'})$ where

$$\gamma_j^l = \arg\min_{\gamma\in\mathbb{R}^n} \sum_{i=1}^n h(f^l(x_i), j)\|\phi(x_i) - \sum_{i'=1}^n \gamma_{i'}\phi(x_{i'})\|^2. \tag{3}$$

Equation (3) has a simple solution:

$$p_j = \frac{\sum_{i=1}^n h(f^l(x_i), j)\phi(x_i)}{\sum_{i=1}^n h(f^l(x_i), j)} \tag{4}$$

which is the weighted mean of the $(\phi(x_i))_i$. As a consequence, equation (3) is the representation step of a batch SOM computed in the feature space $\mathcal{H}$. We will call this algorithm *kernel batch SOM*:

Algorithm 3: Kernel batch SOM

(1) For all $j = 1, \dots, M$ and all $i = 1, \dots, n$, initialize γ_{ji}^0 randomly in $\mathbb{R}$;

(2) For $l = 1, \dots, L$, do

(3) **assignment step**: for all $i = 1, \dots, n$, x_i is assigned to the neuron, $f^l(x_i)$ which has the closest prototype:

$$f^l(x_i) = \arg\min_{j=1,\dots,M} \|\phi(x_i) - p_j^{l-1}\|$$

(4) **representation step**: for all $j = 1, \dots, M$, the prototype p_j is recomputed:

$$\gamma_j^l = \arg\min_{\gamma\in\mathbb{R}^n} \sum_{i=1}^n h(f^l(x_i), j)\|\phi(x_i) - \sum_{i'=1}^n \gamma_{i'}\phi(x_{i'})\|^2$$

End for.

where, as shown in (4), the representation step simply reduces to

$$\gamma_{ji}^l = \frac{h(f^l(x_i), j)}{\sum_{i'=1}^n h(f^l(x_{i'}, j))}.$$

Like in the on-line version, the assignment is run by directly using the kernel without explicitly defining ϕ and $\mathcal{H}$: in fact, and all $x \in \{x_i, \dots, x_n\}$, it leads to the minimization on $j \in \{1, \dots, M\}$ of $\sum_{i,i'=1}^n \gamma_{ji}\gamma_{ji'}k(x_i, x_{i'}) - 2\sum_{i=1}^n \gamma_{ji}k(x, x_i)$. Then, the kernel batch SOM is simply a batch kernel SOM performed in a relevant space so it shares its consistancy properties [7].

Finally, we conclude that the dissimilarity SOM described in section 2.2 can be seen as the restriction of a kernel batch SOM to the case where the prototypes are elements of the original data set. Formally, dissimilarity SOM *is* the batch kernel SOM for which the feature space is not Hilbertian but discrete.

3 Application to graphs

The fact that the prototypes are defined in the feature space $\mathcal{H}$ from the original data $\{x_1, \dots, x_n\}$ allows to apply the algorithms described in section 2 to a wide variety of data, as long as a kernel can be defined on the original set $\mathcal{G}$ (for which no vector structure is needed). In particular, this algorithm can be used to cluster the vertices of a weighted graph into homogeneous proximity groups using the graph structure only, without any assumption on the vertices set. The problem of clustering the vertices of a graph is of a great interest, for instance as a tool for understanding the organization of social networks ([3]). This approach has already been tested for the dissimilarity SOM on a graph extracted from a medieval database ([22]).

We use in the rest of the paper the following notations. The dataset $\{x_1, \dots, x_n\}$ consists in the vertices of a graph $\mathcal{G}$, with a set of edges in E. Each edge $(x_i, x_{i'})$ has a positive weight $w_{i,i'}$ (with $w_{i,i'} = 0 \Leftrightarrow (x_i, x_{i'}) \notin E$). Weights are assumed to be symmetric ($w_{i,i'} = w_{i',i}$). We call d_i the *degree* of the vertex x_i given by $d_i = \sum_{i'=1}^n w_{i,i'}$.

3.1 The Laplacian and related kernels

In [19], the authors investigate a family of kernels based on regularization of the Laplacian matrix. The Laplacian of the graph is the positive matrix $L = (L_{i,i'})_{i,i'=1,\dots,n}$ such that

$$L_{i,i'} = \begin{cases} -w_{i,i'} & \text{if } i \neq i' \\ d_i & \text{if } i = i' \end{cases}$$

(see [5, 14] for a comprehensive review of the properties of this matrix). In particular, [19] shows how this discrete Laplacian can be derived from the usual Laplacian defined on continuous spaces. Applying regularization functions to the Laplacian, we obtain a family of matrices including

- *Regularized Laplacian*: for $\beta > 0$, $K^\beta = (\mathbb{I}_n + \beta L)^{-1}$;
- and *Diffusion matrix*: for $\beta > 0$, $K^\beta = e^{-\beta L}$;

These matrices are easy to compute for graphs having a few hundred of vertices via an eigenvalue decomposition: their eigensystem is deduced by applying the regularizing functions to the eigenvalues of the Laplacian (the eigenvectors are the same). Moreover, these matrices can be interpreted as regularizing matrices because the norm they induced penalizes more the vectors that vary a lot over close vertices. The way this penalization is taken into account depends on the regularizing function applied to the Laplacian matrix.

Using these regularizing matrices, we can define associated kernel by $k^\beta(x_i, x_{i'}) = K_{ii'}^\beta$. Moreover, the diffusion kernels (see also [14]) can be interpreted as the quantity of energy accumulated after a given time in a vertex if energy is injected at time 0 in the other vertex and if the diffusion is done along the edges. It has then become very popular to summarize both the global structure and the local proximities of a graph (see [21, 18] for applications in computational biology). In [9], the authors investigate the distances induced by kernels of this type in order to rank the vertices of a weighted graph; they compare them to each others and

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

show their good performances compared to standard methods.

3.2 Simulations

In order to test the 3 algorithms presented in section 2 for clustering the vertices of a graph, we simulated 50 graphs having a structure close to the ones described in [4, 22]. We made them as follow:

- We built 5 complete sub-graphs $(\mathcal{C}_i)_{i=1,\dots,5}$ (cliques) having $(n_i)_{i=1,\dots,5}$ vertices where n_i are generated from a Poisson distribution with parameter 50;
- For all $i = 1, \dots, 5$, we generated l_i random links between $\mathcal{C}_i$ and vertices of the other cliques: l_i is generated from a uniform distribution on the set $\{1, \dots, 100n_i\}$. Finally, multiple links are removed; thus the simulated graphs are "non-weighted" (i.e. $w_{i,i'} \in \{0, 1\}$).

A simplified version of these types of graphs is shown in Figure 1: we restricted the mean of x_i to 5 and l_i is generated from a uniform distribution on $\{1, \dots, 10n_i\}$ in order to make the visualization possible.

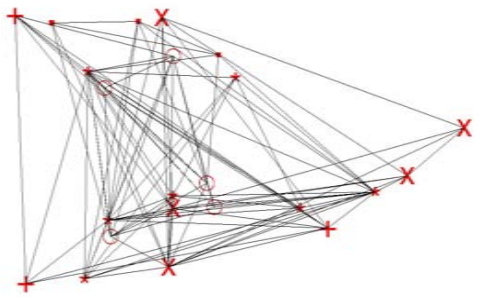

Figure 1: Example of a simulated graph: the vertices of the 5 cliques are represented by different labels (+ □ * x o)

Algorithms 1 to 3 were tested on these graphs by using the diffusion kernel and the dissimilarities (equations (1) and (2)) built from it. The grid chosen is a 3×3 rectangular grid for which the central neuron has a neighborhood of size 2, as illustrated in Figure 2.

We ran all the algorithms until the stabilization is obtained, which leads to:

- 500 iterations for the on-line kernel SOM (algorithm 1);
- 20 iterations for the dissimilarity SOM with both dissimilarities (algorithm 2);
- 10 iterations for the batch SOM (algorithm 3).

Then, to avoid the influence of the initialization step, the algorithms were initialized randomly 10 times. For each graph and each algorithm, the best classification, which minimizes the energy of the final grid, is kept. The computational burden of the algorithms is summarize in Table 1: it gives the total running time for 50 graphs and 10 initializations per graph. Batch kernel SOM is the fastest whereas the on-line kernel SOM is very slow because it needs a high number of iterations to stabilize. For the batch kernel SOM, we initialize the prototypes with random elements of the data set (as in the dissimilarity SOM) in order to obtain a good convergence of the algorithm.

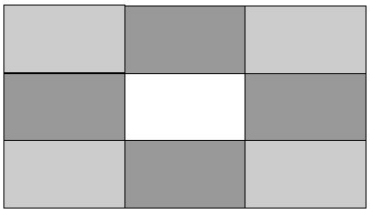

Figure 2: SOM grid used (dark gray is the 1-neighborhood and light gray the 2-neighborhood of the central neuron)

Algorithm	on-line k-SOM	d-SOM	batch k-SOM
Time (min)	260	80	20

Table 1: Computation times

Finally, we tested three parameters for the diffusion kernel ($\beta = 0.1$, $\beta = 0.05$ and $\beta = 0.01$). Higher parameters were not tested because we had numerical instabilities in the computation of the dissimilarities for some of the 50 graphs. In order to compare the classifications obtained by the different algorithms, we computed the following criteria:

- the mean energy (except for the dissimilarity SOM with dissimilarity (1) which does not have a comparable energy). Notice also that the energies computed for different values of β can also not be compared;
- the standard deviation of the energy;
- the mean number of classes found by the algorithm;
- after having associated each neuron of the grid to one of the cliques by a majority vote law, the mean pourcentage of good classified vertices (vertices assigned to a neuron associated to its clique);
- the number of links divided by the number of possible links between 2 vertices assigned to 2 neurons having distance 1 (or 2, 3, 4) between them on the grid.

The results are summarized in Tables 2 to 5 and an example of a classification obtained for the batch kernel SOM for the graph represented in Figure 1 is given in Figure 3.

First of all, we see that the quality of the classification heavily depends on the choice of β. For this application, performances decrease with β, with very bad performances for all the algorithms with the parameter $\beta = 0.01$.

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

	$\beta = 0.1$	$\beta = 0.05$	$\beta = 0.01$
Mean energy	0.10	3.21	196
Sd of energy	0.40	4.7	39
Mean nb of classes	8.04	8.92	9
Mean % of good classif.	79.84	78.28	39.72
% of links for 1-neighborhood	54.5	58.4	51.3
% of links for 2-neighborhood	39.1	40.0	48.0
% of links for 3-neighborhood	34.9	33.1	45.6
% of links for 4-neighborhood	24.2	28.9	43.8

Table 2: Performance criteria for the on-line kernel SOM

	$\beta = 0.1$	$\beta = 0.05$	$\beta = 0.01$
Mean energy	NA	NA	NA
Sd of energy	NA	NA	NA
Mean nb of classes	9	9	9
Mean % of good classif.	77.34	40.49	29.56
% of links for 1-neighborhood	48.6	45.4	52.5
% of links for 2-neighborhood	42.0	45.5	55.8
% of links for 3-neighborhood	38.0	48.1	57.1
% of links for 4-neighborhood	34.8	51.5	57.0

Table 3: Performance criteria for the dissimilarity SOM (dissimilarity (1))

	$\beta = 0.1$	$\beta = 0.05$	$\beta = 0.01$
Mean energy	0.13	3.99	300
Sd of energy	0.55	7.3	65
Mean nb of classes	9	9	9
Mean % of good classif.	77.89	40.69	29.77
% of links for 1-neighborhood	49.6	45.0	52.3
% of links for 2-neighborhood	41.7	45.7	55.8
% of links for 3-neighborhood	36.9	48.0	56.8
% of links for 4-neighborhood	34.0	51.0	58.8

Table 4: Performance criteria for the dissimilarity SOM (dissimilarity (2))

	$\beta = 0.1$	$\beta = 0.05$	$\beta = 0.01$
Mean energy	0.10	3.00	172
Sd of energy	0.38	4.45	35
Mean nb of classes	6.56	7.56	9
Mean % of good classif.	94.34	92.72	32.81
% of links for 1-neighborhood	44.9	48.0	47.6
% of links for 2-neighborhood	37.8	37.6	46.6
% of links for 3-neighborhood	29.1	32.3	48.8
% of links for 4-neighborhood	28.6	28.5	46.4

Table 5: Performance criteria for batch kernel SOM

Then, we can also remark that the performances highly depend on the graph: the standard deviation of the energy is high compared to its mean. In fact, 5 graphs always obtained very bad performances and removing them divides the standard deviation by 20.

Comparing the algorithms to each others, we see that the batch kernel SOM seems to find the best clustering for the vertices of the graph: this is the one for which the mean number of classes found by the algorithm is the closest to the number of cliques (5). It has also the best pourcentage of good classified vertices and the smallest number of links for all neighborhoods, proving that the vertices classified in the same cluster are also frequently in the same clique. Then comes the on-line kernel SOM that suffers from a long computational time and finally, the dissimilarity SOM with slightly better performances for the dissimilarity (2). Comparing the first three Tables, we can say that the performance gain of the on-line kernel SOM is really poor compared to the computational time differences between the algorithms. Moreover, interpretation of the clusters meaning can take benefit of the fact that prototypes are elements of the data set. On the contrary, dissimilarity SOM totally fails in decreasing the number of relevant classes (the mean number of clusters in the final classification is always the biggest possible, 9); this leads to a bigger number of links between two distinct neurons than in the both versions of kernel SOM.

4 Conclusions

We show in this paper that the dissimilarity SOM used with a kernel based dissimilarity is a particular case of a batch kernel SOM. This leads us to the definition of a batch unsupervised algorithm for clustering the vertices of a graph. The simulations made on randomly generated graphs prove that this batch version of kernel SOM is both efficient and fast. The dissimilarity SOM, which is more restricted, is less efficient but still have good performances and produces prototypes that are more easily interpretable. We also emphasized the importance of a good choice of the parameter

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

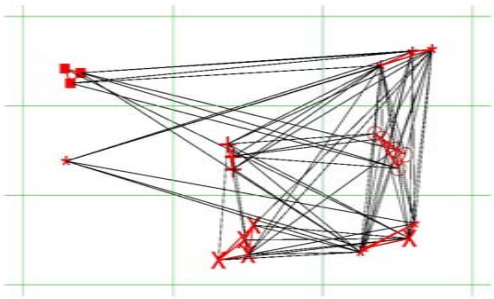

Figure 3: Example of a classification obtained for the batch kernel SOM on graph represented in Figure 1

of the kernel, a problem for which an automatic solution would be very useful in practice.

Acknowledgements

This project is supported by "ANR Non Thématique 2005 : Graphes-Comp". The authors also want to thank the anonymous referees for their helpful comments.

References

[1] C. Ambroise and G. Govaert. Analyzing dissimilarity matrices via Kohonen maps. In *Proceedings of 5th Conference of the International Federation of Classification Societies (IFCS 1996)*, volume 2, pages 96–99, Kobe (Japan), March 1996.

[2] P. Andras. Kernel-Kohonen networks. *International Journal of Neural Systems*, 12:117–135, 2002.

[3] S. Bornholdt and H.G. Schuster. *Handbook of Graphs and Networks - From the Genome to the Internet*. Wiley-VCH, Berlin, 2002.

[4] R. Boulet and B. Jouve. Partitionnement d'un réseau de sociabilité à fort coefficient de clustering. In *7èmes Journées Francophones "Extraction et Gestion des Connaissances"*, pages 569–574, 2007.

[5] F. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

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

[7] M. Cottrell, B. Hammer, A. Hasenfuss, and T. Villmann. Batch and median neural gas. *Neural Networks*, 19:762–771, 2006.

[8] A. El Golli, F. Rossi, B. Conan-Guez, and Y. Lechevallier. Une adaptation des cartes auto-organisatrices pour des données décrites par un tableau de dissimilarités. *Revue de Statistique Appliquée*, LIV(3):33–64, 2006.

[9] F. Fouss, L. Yen, A. Pirotte, and M. Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. In *IEEE International Conference on Data Mining (ICDM)*, pages 863–868, 2006.

[10] T. Graepel, M. Burger, and K. Obermayer. Self-organizing maps: generalizations and new optimization techniques. *Neurocomputing*, 21:173–190, 1998.

[11] T. Kohohen and P.J. Somervuo. Self-Organizing maps of symbol strings. *Neurocomputing*, 21:19–30, 1998.

[12] T. Kohonen. Self-organizing maps of symbol strings. Technical report a42, Laboratory of computer and information science, Helsinki University of technoligy, Finland, 1996.

[13] T. Kohonen. Self-Organizing Maps, 3rd Edition. In *Springer Series in Information Sciences*, volume 30. Springer, Berlin, Heidelberg, New York, 2001.

[14] R.I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

[15] K.W. Lau, H. Yin, and S. Hubbard. Kernel self-organising maps for classification. *Neurocomputing*, 69:2033–2040, 2006.

[16] D. Mac Donald and C. Fyfe. The kernel self organising map. In *Proceedings of 4th International Conference on knowledge-based intelligence engineering systems and applied technologies*, pages 317–320, 2000.

[17] B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear components analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[18] B. Schölkopf, K. Tsuda, and J.P. Vert. *Kernel methods in computational biology*. MIT Press, London, 2004.

[19] A.J. Smola and R. Kondor. Kernels and regularization on graphs. In M. Warmuth and B. Schölkopf, editors, *Proceedings of the Conference on Learning Theory (COLT) and Kernel Workshop*, 2003.

[20] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[21] J.P. Vert and M. Kanehisa. Extracting active pathways from gene expression data. *Bioinformatics*, 19:238ii–244ii, 2003.

[22] N. Villa and R. Boulet. Clustering a medieval social network by SOM using a kernel based distance measure. In M. Verleysen, editor, *Proceedings of ESANN 2007*, pages 31–36, Bruges, Belgium, 2007.

[23] H. Yin. On the equivalence between kernel self-organising maps and self-organising map mixture density networks. *Neural Networks*, 19:780–784, 2006.