Bagged Kernel SOM

Jérôme Mariette[1], Madalina Olteanu[2], Julien Boelaert[2] and Nathalie Villa-Vialaneix[1,2,3]

[1] INRA, UR 0875 MIA-T, BP 52627, 31326 Castanet Tolosan cedex, France
[2] SAMM, Université Paris 1 Panthéon-Sorbonne,
90 rue de Tolbiac, 75634 Paris cedex 13, France
[3] UPVD, 52 avenue Paul Alduy, 66860 Perpignan cedex 9, France
jerome.mariette@toulouse.inra.fr, madalina.olteanu@univ-paris1.fr,
julien.boelaert@gmail.com, nathalie.villa@toulouse.inra.fr

Abstract. In a number of real-life applications, the user is interested in analyzing non vectorial data, for which kernels are useful tools that embed data into an (implicit) Euclidean space. However, when using such approaches with prototype-based methods, the computational time is related to the number of observations (because the prototypes are expressed as convex combinations of the original data). Also, a side effect of the method is that the interpretability of the prototypes is lost. In the present paper, we propose to overcome these two issues by using a bagging approach. The results are illustrated on simulated data sets and compared to alternatives found in the literature.

1 Introduction

In a number of real-life applications, the user is interested in analyzing data that are non described by numerical variables as is standard. For instance, in social network analysis, the data are nodes of a graph which are described by their relations to each others. Self-Organizing Maps (SOM) and other prototype based algorithms have already been extended to the framework of non numerical data, using various approaches. One of the most promising one is to rely on kernels to map the original data into an (implicit) Euclidean space in which the standard SOM can be used [1,2,3]. A closely related approach, called "relational SOM" [4,5], extends this method to dissimilarity data which are pertaining to a pseudo-Euclidean framework, as demonstrated in [4]. Further, in [6], we addressed the issue of using several sources of (possibly non numeric) data by combining several kernels. The combination of kernels is made optimal with a stochastic gradient descent scheme that is included in the on-line version of the SOM algorithm.

However, while able to handle non Euclidean data, that can eventually come from different sources, these approaches suffer from two drawbacks: as pointed out in [7], when the data set is very large, the computational time of such approaches can be prohibitive. Indeed, prototypes are expressed as convex combinations of the original data and are thus expressed with a number of coefficients equal to the number of observations in the data set. Also, adding an extra gradient descent step to optimize the kernel combination requires to increase the

number of iterations of the algorithm, which yields to an augmented computational time. The second drawback is emphasized in [8]: as the prototypes are expressed as a convex combination of the original data, they are no longer given as explicit representative points in the data space and the interpretability of the model is lost.

In the present paper, we propose to overcome these two issues by using a bagging approach in which only a small subset of the original data set is used. The results coming from several bags are combined to select the most representative observations that are then utilized to define the prototypes in a final map. This approach is both sparse (the resulting map is based on a small subset of observations only), fast and parallelizable, which makes it an interesting approach to analyze large samples. The rest of the paper is organized as follow: Section 2 describes the method and its relations to previous approaches in the literature. Section 3 provides the analysis of the results obtained on simulated data sets and on a real-world data set which is a graph.

2 Method

2.1 A brief description of the kernel SOM approach

Let us suppose that we are given input data, $(x_i)_{i=1,\dots,n}$ taking values in an arbitrary space $\mathcal{G}$. When $\mathcal{G}$ is not Euclidean, a solution to handle the data set $(x_i)_i$ with standard learning algorithms is to suppose that a *kernel* is known, i.e., a function $K : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ which is symmetric ($\forall\, z, z' \in \mathcal{G}$, $K(z, z') = K(z', z)$) and positive ($\forall\, N \in \mathbb{N}$, $\forall\, (z_j)_{j=1,\dots,N} \subset \mathcal{G}$ and $\forall\, (\alpha_j)_{j=1,\dots,N} \subset \mathbb{R}$, $\sum_{j,j'} \alpha_j \alpha_{j'} K(z_j, z_{j'}) \geq 0$). When such conditions are fulfilled, the so-called kernel defines a dot product in an underlying Hilbert space: more precisely, there exists a Hilbert space $(\mathcal{H}, \langle ., . \rangle_{\mathcal{H}})$, called the *feature space*, and a function $\phi : \mathcal{G} \to \mathcal{H}$, called the *feature map*, such that

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

(see [9]). Hence using the kernel as a mean to measure similarities between data yields to implicitly rely on the Euclidean structure of $\mathcal{H}$. Many algorithms have been *kernelized*, i.e., modified to handle (possibly non vectorial) data described by a kernel. In particular, the general framework of kernel SOM is described in [1,2,3]. In this framework, as in the standard SOM, the data are clustered into a low dimensional grid made of U neurons, $\{1, \dots, U\}$ and these neurons are related to each other by a neighborhood relationship on the grid, h. Each neuron is also represented by a prototype p_u (for some $u \in \{1, \dots, U\}$) but unlike standard numerical SOM, this prototype does not take value in $\mathcal{G}$ but in the previously defined feature space $\mathcal{H}$. Actually, each prototype is expressed as a convex combination of the image of the input data by the feature map:

$$p_u = \sum_{i=1}^{n} \gamma_{ui} \phi(x_i), \qquad \text{with} \qquad \gamma_{ui} \geq 0 \text{ and } \sum_i \gamma_{ui} = 1.$$

In the on-line version of the algorithm, two steps are iteratively performed:

– **an affectation step** in which an observation x_i is picked at random and affected to its closest prototype using the distance induced by K:

$$f(x_i) = \arg\min_u \|p_u - \phi(x_i)\|^2_{\mathcal{H}},$$

where $\|p_u - \phi(x_i)\|^2_{\mathcal{H}} = K(x_i, x_i) - 2\sum_l \gamma_{uj} K(x_i, x_j) + \sum_{jj'} \gamma_{uj}\gamma_{uj'} K(x_j, x_{j'})$.

– **a representation step** in which the prototypes are updated according to their value at the previous step t and to the observation chosen in the previous step. A gradient descent-like step is used for this update:

$$\forall\, u = 1, \ldots, U, \qquad \gamma_u^{t+1} = \gamma_u^t + \mu(t) h^t(f(x_i), u)\left(\delta_i^n - \gamma_u^t\right)$$

where δ_i^n is the n-dimensional vector in which only entries indexed by i is non zero and equal to 1, $\mu(t) \sim 1/t$ is a vanishing coefficient and, usually, the neighborhood relationship h^t also vanishes with until being restricted to the neuron itself.

Note that this algorithm has also been extended to the case where the observations are described by several kernels, each corresponding to one particular type of data, in the *multiple Kernel SOM* algorithm [6]. In this algorithm, an additional gradient descent step is added to the algorithm to tune the respective contribution of each kernel in an optimal way.

2.2 Ensemble of SOMs

Despite their generalization properties to complex data, kernel SOM and related methods are not well-suited for large data sets since the algorithms generally require the storage of the entire Gram matrix and since the prototypes are expressed as convex combinations of the input data and thus have a very high dimension. Another important drawback of the prototype representation is that, being expressed as a very large linear correlation of the mapped input data, they are not easy to interpret. Indeed, for non vectorial data, such as e.g., nodes in a graph whose similarities can be described by several types of kernels (see [10]), there is no way to describe the prototypes in terms of an object in the input space (here, the graph). As prototypes are commonly used to deciphter the clusters' meaning, one of the main interesting feature of the SOM algorithm is lost in the process, as pointed out in [8].

Several techniques have been proposed in the literature to overcome the dimensionality issues, which can be adapted to the kernel SOM framework: some are related to sparse representations and some to bootstraping and bagging. In [4], the large size of the data set is handled using "patch clustering", which is particularly suited for streaming data but can also be used to handle large dimensional data. The initial data set is randomly split into several patches, $\mathcal{P}_b$ and the algorithm processes each patch iteratively. At step b, the b-th patch is clustered until convergence. Each of the resulting prototypes is approximated by the closest P input data points. During the next step, the index set of the

P-approximations of all prototypes and the index set of the next patch $\mathcal{P}_{b+1}$ are put together into the extended patch $\mathcal{P}^\star_{b+1}$ and the clustering process is performed on all the observations indexed by $\mathcal{P}^\star_{b+1}$. This is iterated until all patches are clustered. This approach leads to good clustering results, however it is not parallelizable and the algorithm may be sensitive to the order in which patches are processed. Another technique for handling large data sets is to use bootstrap and bagging. In [11], bagging is applied to a batch version of the SOM algorithm for numerical data in a semi-supervised context. The prototypes of the map trained on the first bag are initialized to lie in the first principal component and each trained map is used to initialize the subsequent map for the subsequent bag. This procedure reduces the dimensionality and improves the classification error, but it is not parallelizable. Alternatively, [12,13] propose by combining SOM based on separate bootstrap samples with a fusion of their prototypes. These approach, which can be used in parallel are however only valid if the prototypes are expressed on the same representation space, which is not directly generalizable when using kernel SOM in which prototypes are directly expressed with the bootstrap sample.

2.3 Bagged kernel SOM

Our proposal is to use a bagging approach that is both parallelizable and sparse. Bagging uses a large number of small sub-samples, all randomly chosen, to select the most relevant observations: B subsets, $(\mathcal{S}_b)_b$ each of size $n_B \ll n$, are built, at random, within the original data set $\{x_1, \ldots, x_n\}$. Using the on-line algorithm described in [14], a map with U neurons is trained, which results in the prototypes $p_u^b = \sum_{x_i \in \mathcal{S}_b} \gamma_{ui}^b \phi(x_i)$ where ϕ is the feature map associated with the kernel K. The most representative observations are chosen as the first P largest weights for each prototype: $\forall\, u = 1, \ldots, U$,

$$\mathcal{I}_u^b := \left\{ x_i : \ \gamma_{ui} \text{ is one of the first } P \text{ largest weights among } (\gamma_{uj}^b)_{x_j \in \mathcal{S}_b} \right\},$$

and $\mathcal{I}_b = \cup_u \mathcal{I}_u^b$. Alternative methods to select the most interesting prototypes are reported in [8]; the one we chose is referred in this paper as the *K-convex hull* but it would be interesting to test other methods for selecting the most interesting observations.

Then, the number of times each observation $(x_i)_{i=1,\ldots,n}$ is selected in one sub-sample is computed: $\mathcal{N}(x_i) := \sharp\{b : \ x_i \in \mathcal{I}_b\}$ which is finally used as a quality criterion to select the most important variables which are the first $P \times U$ observations with the largest values for $\mathcal{N}(x_i)$:

$$\mathcal{S} := \{x_i : \mathcal{N}(x_i) \text{ is one of the first } PU \text{ largest numbers among } (\mathcal{N}(x_j))_{j \geq n}\}.$$

A final map is then trained with the selected observations in $\mathcal{S}$ which has prototypes expressed as $p_u = \sum_{x_i \in \mathcal{S}} \gamma_{ui} \phi(x_i)$. The final classification for all observations $(x_i)_{i=1,\ldots,n}$ is deduced from these prototypes by applying the standard affectation rule: $\mathcal{C}(x_i) := \arg\min_{u=1,\ldots,U} \|p_u - \phi(x_i)\|^2$ where $\|p_u - \phi(x_i)\|^2_{\mathcal{H}}$ is

Algorithm 1 Multiple online kernel SOM

Initialize for all $i = 1, \ldots, n$, $\mathcal{N}(x_i) \leftarrow 0$
for $b = 1 \rightarrow B$ **do**
  Sample randomly with replacement n_B observations in $(x_i)_{i=1,\ldots,n}$ **return** $\mathcal{S}_b$
  Perform **kernel SOM** with $\mathcal{S}_b$ **return** prototypes $(p_u^b)_u \sim (\gamma_{ui}^b)_{ui}$
  for $u = 1 \rightarrow U$ **do**
    Select the P largest $(\gamma_{ui}^b)_{x_i \in \mathcal{S}_b}$ **return** $\mathcal{I}_u^b$ (set of the observations corresponding to the selected γ_{ui}^b)
  end for
  for $i = 1 \rightarrow n$ **do**
    if $x_i \in \cup_u \mathcal{I}_u^b$ **then**
      $\mathcal{N}(x_i) \leftarrow \mathcal{N}(x_i) + 1$
    end if
  end for
end for
Select the PU observations x_i corresponding to the largest $\mathcal{N}(x_i)$ **return** $\mathcal{S}$
Perform **kernel SOM** with $\mathcal{S}$ **return** prototypes $(p_u)_u \sim (\gamma_{ui})_{u=1,\ldots,U, x_i \in \mathcal{S}}$ and classification $(f(x_i))_{x_i \in \mathcal{S}}$
Affect $(x_i)_{x_i \notin \mathcal{S}}$ with

$$f(x_i) := \arg\min_u \|\phi(x_i) - p_u\|_{\mathcal{H}}^2$$

return final classification $(f(x_i))_{i=1,\ldots,n}$ and sparse prototypes $(p_u)_u \sim (\gamma_{ui})_{u=1,\ldots,U, x_i \in \mathcal{S}}$

computed using K, as described in Section 2.1. The algorithm is described in Algorithm 1.

Note that, strictly speaking, only the sub-kernels $\mathbf{K}_{\bar{\mathcal{S}},\mathcal{S}} := (K(x_i, x_j))_{i \notin \mathcal{S}, j \in \mathcal{S}}$ and $\mathbf{K}_{\mathcal{S}} = (K(x_j, x_j'))_{j,j' \in \mathcal{S}}$ are required to perform the final affectation step because the closest prototype does not depend on the term $K(x_i, x_i)$ and thus the affectation step for $(x_i)_{i \notin \mathcal{S}}$ can be performed by computing:

$$-2\mathbf{K}_{\bar{\mathcal{S}},\mathcal{S}}\gamma + \mathbf{1}_{|\bar{\mathcal{S}}|} \left[\mathrm{Diag}\left(\gamma^T \mathbf{K}_{\mathcal{S}} \gamma\right)\right]^T,$$

where $\gamma = (\gamma_{ui})_{u=1,\ldots,U, i \in \mathcal{S}}$ and $\mathbf{1}_{|\bar{\mathcal{S}}|}$ if the vector with all entries equal to 1 and having length the number of elements in $\bar{\mathcal{S}} = \{x_i : \ x_i \notin \mathcal{S}\}$.

The complexity of the approach is $\mathcal{O}(Un_B^2 B + U^2 P)$, compared to the direct approach which has a complexity equal to $\mathcal{O}(Un^2)$. Hence, the computational time is reduced as long as $Bn_B^2 + U^2 P < n^2$ and is even more reduced if the B sub-SOMs are performed in parallel. Usually, B is chosen to be large, n_B is small compared to n and P is only a few observations to obtain sparse representations of the prototypes. However, the computational times are not directly comparable since the bagged approach can be performed in parallel, unlike the direct approach or the patch SOM.

3 Applications

Bagged kernel SOM on simulated data

First, a simple simulated dataset with 5000 observations is considered. The observations are randomly generated in $[0,1]^{20}$ and are then mapped onto a 5×5 grid using the kernel SOM algorithm with a Gaussian kernel. Several algorithms are compared with varying parameters:

- the **patch SOM** with different numbers of patches (250, 375, 500, 1 000) and different values for P (2, 5, 10, 15, 20, 30 and 50). A last kernel SOM was trained with the selected observations to make the results (based on P selected observations) comparable with those of the patch SOM;
- the **bagged SOM** with different values for n_B (5%, 7.5%, 10% and 20% of the original data set size) and for B (500 and 1000) and the same values for P as with the patch SOM;
- a **full kernel SOM** used on the whole data set and aimed at being the reference method.

Figure 1 gives the quantization and topographic [15] errors of the resulting maps versus the value of P. In this figure, two classical quality criteria for SOM results

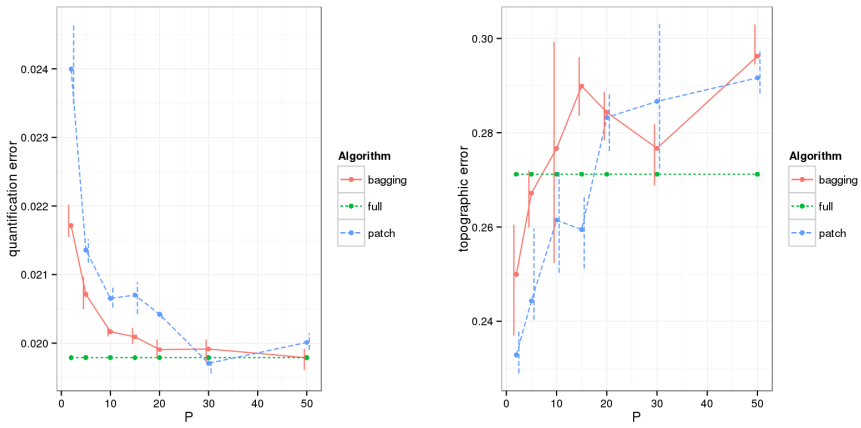

Fig. 1. Evolution of the quantization (left) and topographic (right) errors versus P. Error bars indicates the first and last quantiles and dots the average values over all simulations.

are used: the quantization error (which assesses the quality of the clustering) and the topographic error (which assesses the quality of the organization; see [15]). In some cases, the results can be even better than the full kernel SOM. Considering the bootstrap version, the performances are consistent with the full kernel SOM (for about $P \sim 5-10$, which corresponds to using only 250 observations at most, instead of 5000, to represent the prototypes).

The analysis of the other parameters of the algorithm (bag size n_B and number of bootstrap samples B) does not show any particular feature. This is

explained because the final clustering is obtained from the PU most representative observations and thus P has a much greater impact on the performances than, e.g., n_B.

Application to ego-facebook© network

The bagging method is then applied to one of the ego-facebook© networks described in [16][4]. The data used in this section are the ones extracted from the network number 107: the largest connected component of the facebook© network was extracted, which contained 1 034 nodes. This section presents the comparaison of bagged SOM and standard SOM to map the nodes of the graph onto a two-dimensional grid (having sizes 10×10). As explained in [3,17], using such mappings can provide a simplified representation of the graph, which is useful for the user to help him or her understand its macro-structure before focusing more deeply on some chosen clusters. The kernel used to compute similarities between nodes in the facebook© network was the *commute time kernel*, [18]. If the graph is denoted by $\mathcal{G} = (V, E, W)$, with $V = \{x_1, \ldots, x_n\}$ the set of vertices, E the set of edges which is a subset of $V \times V$ and W a weight matrix (a symmetric matrix with positive entries and null diagonal), the commute time kernel is the generalized inverse of the graph Laplacian, L, which is: $L_{ij} = \begin{cases} d_i & \text{if } i = j \\ -W_{ij} & \text{otherwise} \end{cases}$ where $d_i = \sum_j W_{ij}$ is the degree of node x_i. As explained in [19], the Laplacian is closely related to the graph structure and thus, it is not surprising that a number of kernel has been derived from this matrix [10]. As shown in [18], the commute kernel yields to a simple similarity interpretation because it computes the average time needed for a random walk on the graph to reach a node from another one.

Different approaches were compared: (i) the standard kernel SOM (on-line version), using all available data; (ii) the bagged kernel SOM, as described in Section 2, with $B = 1000$ bootstrap sample, $n_B = 200$ in each sample and $P = 3$ observations selected per prototype and (iii) a standard kernel SOM trained with an equivalent number of randomly chosen observations. The relevance of the results was assessed using different quality measures. Some quality measures were related to the quality of the map (quantification error and topographic error) and some were related to a ground truth: some of the nodes have been indeed labeled by users to belong to one "list" (as named by facebook©). We confronted these groups to the clusters obtained on the map calculating (i) the average node purity (i.e., the mean over all clusters of the maximal proportion of one list in a given cluster; only individuals belonging to one list were used to compute this quality measure) and (ii) the normalized mutual information [20] (also restricted to individuals belonging to one list only) and also to the graph structure using the modularity [21], which is a standard quality measure for node clustering.

The results are summarized in Table 1. Surprisingly, the maps trained with a reduced number of samples (bagged K-SOM and random K-SOM) obtain better quality measures than the map trained with all samples. Using a bootstraping

[4] available at http://snap.stanford.edu/data/egonets-Facebook.html

	Quantification Error (×100)	Topographic Error	Node Purity	Normalized Mutual Information	Modularity
bagged K-SOM	7.66	4.35	89.65	70.10	0.47
full K-SOM	9.06	5.22	86.53	53.79	0.34
random K-SOM	8.08	6.09	87.26	60.79	0.40

Table 1. Quality measures for different versions of kernel SOM (standard using all data, bagged, standard using randomly selected data) on facebook© data

approach to select the relevant observations also significantly improves all quality measures as compared to a random choice with the same number of observations. The results obtain with the bagged SOM are displayed in Figures 2 and 3 (from, respectively, the map and the network points of view). They show that the nodes are mainly dispatched into three big clusters, which correspond each to approximately only one "list", as defined by the user. The results provided

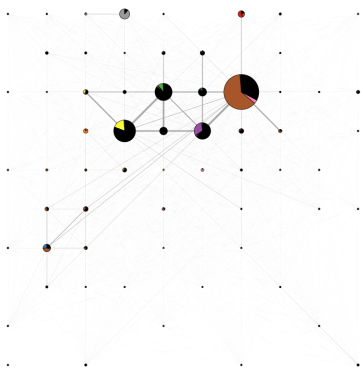

Fig. 2. Simplified representation of the facebook© network projected on the map resulting from bagged K-SOM. The circle sizes are proportional to the number of nodes classified in the cluster and the edge width are proportional to the number of edges between the nodes in the two clusters. Colors correspond to the proportion of user-defined lists (black is used for "no list").

with the K-SOM using all the data tend to provide smaller communities and to scatter the biggest lists on the map. Using this approach, it is however hard to conclude if interpretability has been increased (i.e., if the selected observations used for training are representative of their cluster) as, in Figure 3, they do not seem to have a particularly high degree or centrality.

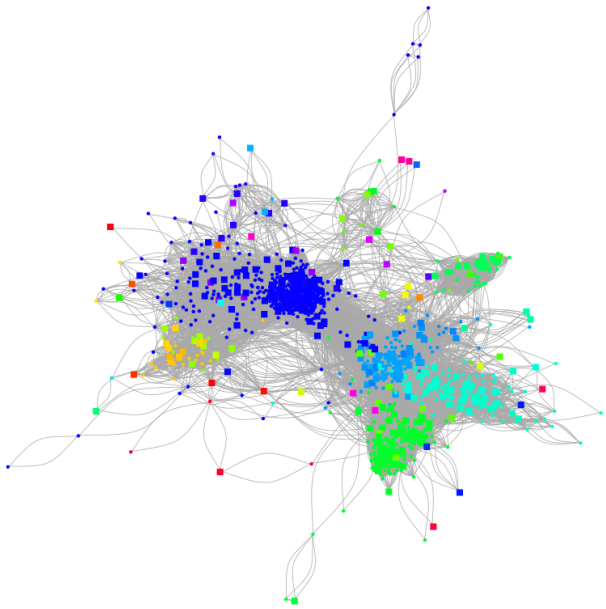

Fig. 3. The facebook© network represented with a force-directed placement algorithm [22]. Colors represent the clusters on the map and selected nodes used to train the map are represented by squares (instead of circles)

4 Conclusion

This paper presents a parallelizable bagged approach which results in a reduced computational time and a sparse representation of prototypes for kernel SOM. The simulations show good performances and only a small loss of accuracy which is compensated by a faster computational time. Obtained prototypes are also easier to interpret, as based on a smaller number of observations.

References

1. 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. (2000) 317–320
2. Lau, K., Yin, H., Hubbard, S.: Kernel self-organising maps for classification. Neurocomputing **69** (2006) 2033–2040
3. Boulet, R., Jouve, B., Rossi, F., Villa, N.: Batch kernel SOM and related laplacian methods for social network analysis. Neurocomputing **71**(7-9) (2008) 1257–1273
4. Hammer, B., Hasenfuss, A.: Topographic mapping of large dissimilarity data sets. Neural Computation **22**(9) (September 2010) 2229–2284
5. Olteanu, M., Villa-Vialaneix, N.: On-line relational and multiple relational SOM. Neurocomputing (2013) Forthcoming.

6. Olteanu, M., Villa-Vialaneix, N., Cierco-Ayrolles, C.: Multiple kernel self-organizing maps. In Verleysen, M., ed.: XXIst European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Bruges, Belgium, i6doc.com (2013) 83–88
7. 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)
8. Hofmann, D., Hammer, B.: Sparse approximations for kernel learning vector quantization. In Verleysen, M., ed.: Proceedings of XXIst European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Bruges, Belgium, i6doc.com (2013) 549–554
9. Aronszajn, N.: Theory of reproducing kernels. Transactions of the American Mathematical Society **68**(3) (1950) 337–404
10. Smola, A., Kondor, R.: Kernels and regularization on graphs. In Warmuth, M., Schölkopf, B., eds.: Proceedings of the Conference on Learning Theory (COLT) and Kernel Workshop. Lecture Notes in Computer Science (2003) 144–158
11. Petrakieva, L., Fyfe, C.: Bagging and bumping self organising maps. Computing and Information Systems Journal **9** (2003) 69–77
12. Vrusias, B., Vomvoridis, L., Gillam, L.: Distributing SOM ensemble training using grid middleware. In: Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN 2007). (2007) 2712–2717
13. Baruque, B., Corchado, E.: Fusion methods for unsupervised learning ensembles. Volume 322 of Studies in Computational Intelligence. Springer (2011)
14. Villa, N., Rossi, F.: A comparison between dissimilarity SOM and kernel SOM for clustering the vertices of a graph. In: 6th International Workshop on Self-Organizing Maps (WSOM), Bielefield, Germany, Neuroinformatics Group, Bielefield University (2007)
15. Polzlbauer, G.: Survey and comparison of quality measures for self-organizing maps. In Paralic, J., Polzlbauer, G., Rauber, A., eds.: Proceedings of the Fifth Workshop on Data Analysis (WDA'04), Sliezsky dom, Vysoke Tatry, Slovakia, Elfa Academic Press (2004) 67–82
16. McAuley, J., Leskovec, J.: Learning to discover social circles in ego networks. In: NIPS Workshop on Social Network and Social Media Analysis. (2012)
17. Rossi, F., Villa-Vialaneix, N.: Optimizing an organized modularity measure for topographic graph clustering: a deterministic annealing approach. Neurocomputing **73**(7-9) (2010) 1142–1163
18. Fouss, F., Pirotte, A., Renders, J., Saerens, M.: Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. IEEE Transactions on Knowledge and Data Engineering **19**(3) (2007) 355–369
19. von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing **17**(4) (2007) 395–416
20. Danon, L., Diaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. Journal of Statistical Mechanics (2005) P09008
21. Newman, M., Girvan, M.: Finding and evaluating community structure in networks. Physical Review, E **69** (2004) 026113
22. Fruchterman, T., Reingold, B.: Graph drawing by force-directed placement. Software, Practice and Experience **21** (1991) 1129–1164