A Self Organizing Map for dissimilarity data[0]

Aïcha El Golli[1,2], Brieuc Conan-Guez[1,2], and Fabrice Rossi[1,2,3]

[1] Projet AXIS, INRIA-Rocquencourt Domaine De Voluceau,
BP 105 Bâtiment 18
78153 Le Chesnay Cedex, France
aicha.el_golli@inria.fr
brieuc.conan-guez@inria.fr
[2] CEREMADE, UMR CNRS 7534, Université Paris Dauphine
rossi@ufrmd.dauphine.fr
[3] Up to date contact informations for Fabrice Rossi are avaible at
http://apiacoa.org/

Summary. Treatment of complex data (for example symbolic data, semi-structured data, or functional data) can not be easily done by clustering methods based on calculating the center of gravity. We present in this paper an extension of self organizing maps to dissimilarity data. This extension allows to apply this algorithm to numerous type of data in a convenient way.

1 Introduction

The Kohonen Self Organizing Map (SOM) introduced by Professor Kohonen (see Kohonen (1997)) is an unsupervised neural network method which has both clustering and visualization properties. It can be considered as an algorithm that maps a high dimensional data space, $\mathbb{R}^p$, to lattice space which usually has a lower dimension, generally 2 and is called a Map. This projection enables a partition of the inputs into "similar" clusters while preserving their topology. Its most similar predecessors are the k-means (see MacQueen (1967)) algorithm and the dynamic clustering method (see Diday et al. (1989)), which operate as a SOM without topology preservation and so without easy visualization.

In data analysis, new forms of complex data have to be considered, most notably structured data (data with an internal structure such as intervals data, distributions, functional data, etc) and semi-structured data (trees, XML documents, SQL queries, etc.). In this context, classical data analysis based on calculating the center of gravity can not be used because inputs are not

[0] Published in IFCS'2004 Proceedings.
Available at http://apiacoa.org/publications/2004/som-ifcs04.pdf

$\mathbb{R}^p$ vectors. In order to solve this problem, several methods can be considered according to the type of data (for example recoding techniques for symbolic data (see de Reyniès (2003)) or projection operators for functional data (see Ramsay and Silverman (1997))). However, those methods are not fully general and an adaptation of every data analysis algorithm to the resulting data is needed.

We propose in this article an adaptation of the SOM to dissimilarity data as an alternative solution. Indeed, Kohonen's SOM is based on the notion of center of gravity and unfortunately, this concept is not applicable to many kind of complex data, especially semi-structured data. Our goal is to modify the SOM algorithm to allow its implementation on dissimilarity measures rather than on raw data. With this alternative only the definition of a dissimilarity for each type of data is necessary to apply the method and so treat complex data.

The paper is organized as follows: we first recall the SOM algorithm in its batch version. Then we describe our adaptation. We conclude the paper by experiments on simulated and real world data.

2 Self-Organizing Map (SOM)

2.1 Introduction

Kohonen's SOM is used nowadays through numerous domains and has been successfully applied in numerous applications. It is a very popular tool used for visualizing high dimensional data spaces. SOM can be considered as doing vector quantization and/or clustering while preserving the spatial ordering of the input data reflected by implementing an ordering of the codebook vectors (also called prototype vectors, cluster centroids or **referent vectors**) in a one or two dimensional output space. The SOM consists of neurons organized on a regular low-dimensional grid, called the map. More formally, the map is described by a graph (C, Γ). C is a set of m interconnected neurons having a discrete topology defined by Γ. For each pair of neurons (c, r) on the map, the distance $\delta(c, r)$ is defined as the shortest path between c and r on the graph. This distance imposes a neighborhood relation between neurons (see figure 1 for an example). Each neuron c is represented by a p-dimensional referent vector $w_c = \{w_c^1, ..., w_c^p\}$, where p is equal to the dimension of the input vectors. The number of neurons may vary from a few dozen to several thousand depending on the application.

The SOM training algorithm resembles k-means (see MacQueen (1967)). The important distinction is that in addition to the best matching referent vector, its neighbors on the map are updated: the region around the best matching vector is stretched towards the training sample presented. The end result is that the neurons on the grid become ordered: neighboring neurons have similar referent vectors.

The SOM takes as its input a set of labeled sample vectors and gives as output an array of neurons with the input vectors labels attached to these neurons. Let n be the number of sample vectors $z_i \in \mathbb{R}^p$, $i = 1, 2, ..., n$, where each sample vector z_i is identified by a label.

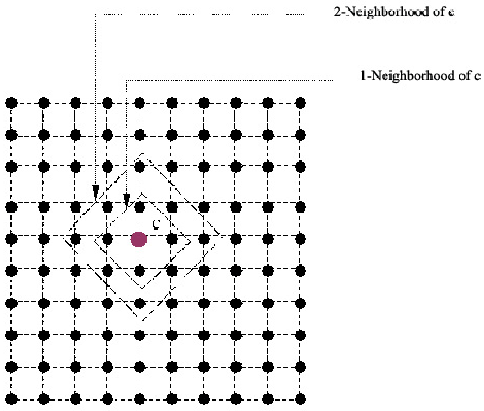

Fig. 1. Discrete topology of a two dimensional topological map (10*10 neurons), each point represents a neuron. 1-neighborhood and 2-neighborhood of neuron c

2.2 Batch training algorithm

The batch training algorithm is an iterative algorithm in which the whole data set (noted Ω) is presented to the map before any adjustments are made. In each training step, the data set is partitioned according to the Voronoi regions of the map referent vectors. More formally, we define an affectation function f from $\mathbb{R}^p$ (the input space) to C, that associates each element z_i of $\mathbb{R}^p$ to the neuron whose referent vector is "closest" to z_i (for the Euclidean distance). This function induces a partition $P = \{P_c; c = 1...m\}$ of the set of individuals where each part P_c is defined by: $P_c = \{z_i \in \Omega; f(z_i) = c\}$. This is the **affectation step**. It is quite clear that this step is rather easy to adapt to a dissimilarity setting.

After affectation, a **representation step** is performed. The algorithm updates the referent vectors by minimizing a cost function, noted $E(f, W)$. This function has to take into account the inertia of the partition P, while insuring the topology preserving property. To achieve these two goals, it is necessary to generalize the inertia function of P by introducing the neighborhood notion attached to the map. In the case of individuals belonging to $\mathbb{R}^p$,

this minimization can be done in a straight way. Indeed new referent vectors are calculated as:

$$w_r^{t+1} = \frac{\sum_{i=1}^{n} h_{rc}(t) z_i}{\sum_{i=1}^{n} h_{rc}(t)}$$

where $c = \arg\min_r ||z_i - w_r||$, is the index of the best matching unit of the data sample z_i, $\|.\|$ is the distance mesure, typically the Euclidean distance, and t denotes the time. $h_{rc}(t)$ the neighborhood kernel around the winner unit c. This function is a nonincreasing function of time and of the distance of unit r from the winner unit c. The new referent vector is a weighted average of the data samples, where the weight of each data sample is the neighborhood function value $h_{rc}(t)$ at its winner c. In the batch version of the k-means algorithm, the new referent vectors are simply averages of the Voronoi data sets. Obviously, the representation step is the one that cannot be directly adapted to a dissimilarity setting in which weighted average of data cannot be performed.

3 A batch self organizing map for dissimilarity data

3.1 Principle

The Map for dissimilarity data is described by a graph (C, Γ) exactly as the traditional SOM. The main difference we are not working on $\mathbb{R}^d$ but on an arbitrary set on which a dissimilarity (denoted d) is defined.

The representation space L_c of a neuron c is the set of parts of Ω with a fixed cardinality q: each neuron c is represented by an "***individual referent***" $a_c = \{z_{j_1}, ..., z_{j_q}\}$, and $z_{j_i} \in \Omega$. We denote a the individuals codebook, i.e. the list $a = \{a_c; c = 1, ..., m\}$ of the individual referents of the map. In classical SOM each referent vector evolves in the entire input space $\mathbb{R}^p$. In our approach each neuron has a finite number of representations.

We define a new dissimilarity d^T from $\Omega \times P(\Omega)$ to $\mathbb{R}^+$ by:

$$d^T(z_i, a_c) = \sum_{r \in C} K^T(\delta(c, r)) \sum_{z_j \in a_r} d^2(z_i, z_j)$$

This dissimilarity is based on a kernel positive function, K. This function is such that $\lim_{|\delta| \to \infty} K(\delta) = 0$ and allows to transform the sharp graph distance between two neurons on the map $(\delta(c, r))$ into a smooth distance. K is used to define a family of functions K^T parameterized by T, with $K^T(\delta) = K(\frac{\delta}{T})$. As for the traditional SOM, T is used to control the size of the neighborhood (see Thiria and et Al (2002)): When the parameter T is small, there are few neurons in the neighborhood. A simple example of K^T is defined by $K^T(\delta) = e^{-\frac{\delta^2}{T^2}}$.

During the learning, we minimize the following cost function E by alternating the affectation step and the representation step:

$$E(f,a) = \sum_{z_i \in \Omega} d^T(z_i, a_{f(z_i)}) = \sum_{z_i \in \Omega} \sum_{r \in C} K^T(\delta(f(z_i), r)) \sum_{z_j \in a_r} d^2(z_i, z_j) \quad (1)$$

This function calculates the adequacy between the induced partition by the affectation function and the map referents a.

During the affectation step, the affectation function f affects each individual z_i to the nearest neuron, here in terms of the dissimilarity d^T:

$$f(z_i) = arg \min_{c \in C} d^T(z_i, a_c) \quad (2)$$

This affectation step decreases the E criterion.

During the representation step, we have to find the new individuals codebook a^* that represents the set of observations in the best way in terms of E. This optimization step can be realized independently for each neuron. Indeed, we minimize the m following functions:

$$E_r = \sum_{z_i \in \Omega} K^T(\delta(f(z_i), r)) \sum_{z_j \in a_r} d^2(z_i, z_j) \quad (3)$$

In the classical batch version, this minimization of the E function is immediate because the positions of the referent vectors are the averages of the data samples weighted by the kernel function.

3.2 The Algorithm

Initialization: iteration $k = 0$, choose an initial individuals codebook a^0. Fix $T = T_{max}$ and the total number of iterations N_{iter}

Iteration: At iteration k, the set of individual referents of the previous iteration a^{k-1} is known. Calculate the new value of T:

$$T = T_{max} * (\frac{T_{min}}{T_{max}})^{\frac{k}{N_{iter-1}}}$$

▶ **affectation step:** up date the affectation function f_{a^k} associated to the a^{k-1} codebook. Affecting each individual z_i to the referent as defined in equation (2).

▶ **representation step:** determine the new codebook a^{k*} that minimizes the $E(f_{a^k}, a)$ function (with respect to a) a_c^{k*} is defined from equation (3).

Repeat **Iteration** until $T = T_{min}$

4 Experiments

In all our experiments the representation space L_c of a neuron c is one individual, i.e $q = 1$.

4.1 Simulated data

The data are distributed in $\mathbb{R}^3$ and represent a geometric form of a cylinder. There are 1000 individuals. The input data is a Euclidean distance matrix and the map contains (20*3) neurons. In the following figures we present the training data and the evolution of the map during the training with the proposed algorithm. Figure 2 is the initial random map. In the final map, shown in figure 5, there is a good quantification while preserving the topology.

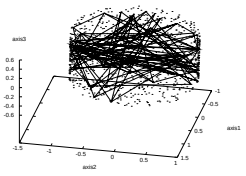

Fig. 2. The initial map (20*3) neurons (random initialization) and the data

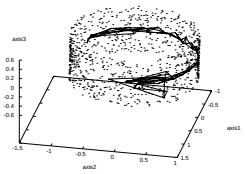

Fig. 3. The map after 50 iterations

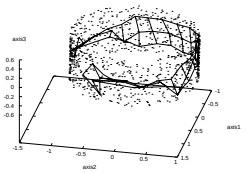

Fig. 4. The map after 100 iterations

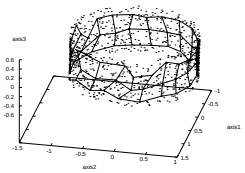

Fig. 5. The final map

4.2 Real world data

The next example is a classification problem of spectrometric data food industry. Each observation is the near infrared absorbance spectrum of a meat sample (finely choped), recorded on a Tecator Infratec Food and Feed Analyser. More precisely, an observation consists in a 100 channel spectrum of absorbances in the wavelength range 850-1050 nm (figure 6). There are 215 spectra in the database. In order to validate the behaviour of the proposed algorithm, we make use of another variable, which measures the fat content of each meat sample (the range of this variable is from 2% to 59%). This variable is indeed deeply linked to the shape of the spectrum, and so the obtained classification should be consistent with the fat value. In the following experiments, all the maps contain (8*2) neurons.

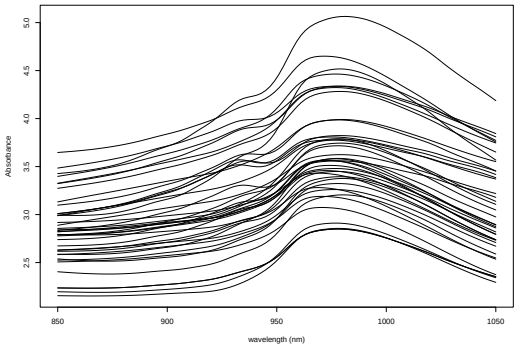

Fig. 6. 40 spectra

In this first experiment, we use the L^2-norm as dissimilarity between spectra : $\| f \|^2 = \int (f(t))^2 dt$. The exact calculation of the integral is approximated thanks to numerical integration (trapezoidal rule). In the figure 7, we show the result obtained by the proposed algorithm. Each square (which is associated to one class) is drawn with an intensity which is calculated in respect to the mean of the fat content of spectra belonging to the class (black for low fat value, and white for high fat value).

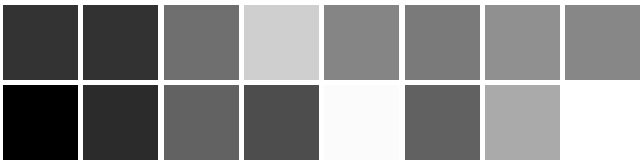

Fig. 7. L^2-norm : the mean of the fat content of each class

Although the obtained classification seems to respect quite well the fat variable (low fat value on the left, and high fat value on the right), the result is not totally satisfactory : we can see that a class with high fat value is just located between two classes with lower fat value. As we can see in the next experiment, this is mainly due to a choice of an inadapted metric. Indeed, in the next experiment, we use as dissimilarity a semi-metric based on the second derivative of spectra : $\| f \|^2_{d^2} = \int (f^{(2)}(t))^2 dt$ (where $f^{(2)}$ denotes the second derivative of f). Ferraty and Vieu (2003) point out that the second derivative of the spectrum is in general more informative than spectrum itself. In order to apply this functional approach, we differentiate each spectrum thanks to a numerical formula (this estimation is consistent, as spectra are very smooth). Each derivative is therefore represented by a vector of 100 components as the original data (figure 8). The integration is done according to the same procedure as the first experiment.

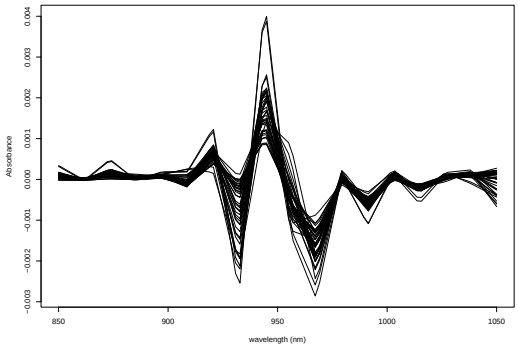

Fig. 8. Second derivatives of the 40 spectra

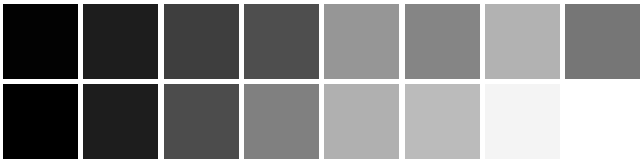

Fig. 9. Second derivative based metric : the mean of the fat content of each class

This time, we can see in the figure 9 that the obtained classification respects perfectly the fat variable. These both examples show that the proposed algorithm depends strongly on the metric : with an appropriate metric, the algorithm behaves in a satisfactory way, as the topology of the map is consistent with the fat variable. Of course, it would have been possible to use a standard SOM to treat this example. In this case, results are in fact quite similar. Our goal in presenting this spectrometric application, is to show both the validity of this approach, and its flexibility.

5 Conclusion

Compare to other clustering methods, self organizing maps allow an easy visualisation of the obtained classification thanks to the preservation of the topology. The extension of SOM(s) to dissimilarity data is straightforward, and gives a very general tool which can be applied to various type of data without any adaptation. The results obtained on both simulated and real world data are satisfactory.

References

de Reyniès, A., 2003. Classification et discrimination en analyse de données symboliques. Ph.D. thesis, Université Paris Dauphine.

Diday, E., Celeux, G., Govaert, G., Lechevallier, Y., H.Ralambondrainy, 1989. Classification automatique des données. DUNOD informatique.

Ferraty, F., Vieu, P., 2003. Curves discriminations: a nonparametric functional approach. Computational Statistics and Data Analysis 44 (1–2), 161–173.

Kohonen, T., 1997. Self-Organisation Maps. Springer Verlag, New York.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In: Proc.of the Fifth Berkeley Symposium on Math., Stat. and Prob. Vol. Vol. 1. pp. 281–296.

Ramsay, J., Silverman, B., June 1997. Functional Data Analysis. Springer Series in Statistics. Springer Verlag.

Thiria, S., et Al, G. D., 2002. Réseaux de neurones méthodologie et applications. Eyrolles, Paris.