Stochastic Self-Organizing Map variants with the \( \text{R} \) package SOMbrero

(Invited Paper)

Nathalie Villa-Vialaneix
MIAT, Université de Toulouse, INRA
BP 52627, 31326 Castanet Tolosan cedex, France
Email: nathalie.villa-vialaneix@inra.fr
Telephone: +33 (0)5 61 28 55 73

Abstract—Self-Organizing Maps (SOM) \cite{1} are a popular clustering and visualization algorithm. Several implementations of the SOM algorithm exist in different mathematical/statistical softwares, the main one being probably the SOM Toolbox \cite{2}. In this presentation, we will introduce an \texttt{R} package, SOMbrero, which implements several variants of the stochastic SOM algorithm. The package includes several diagnosis tools and graphics for interpretation of the results and is provided with a complete documentation and examples.

I. INTRODUCTION

Self-Organizing Maps (SOM), introduced by Teuvo Kohonen \cite{1}, \cite{3}, \cite{4}, are a popular clustering and visualization algorithm. The success of the method is due to its very simple definition as well as its ability to perform, in a single analysis, clustering and visualization by projection of the data onto a small dimensional grid. This is a powerful and easy-to-use method for the exploration of multidimensional data. Originally, SOM has been inspired by neuro-biological learning paradigms which were intended to model some sensory or cognitive processes in which the learning is directed by the experience and the external inputs without supervision. Later, the method has been used in a wide variety of areas (\cite{2} reports more than 7,000 scientific publications related to SOM with applications in industrial, biomedical and financial analyses, for instance).

Several implementations of the SOM algorithm exist in different mathematical/statistical softwares, most on them usable only for numeric data. The main one is probably the SOM Toolbox\textsuperscript{1}, which is released under GPL2 licence and runs in Matlab. The toolbox contains many different functions to learn and interpret SOM and to handle various types of data, including datasets with missing data, symbol strings datasets and relational datasets (i.e., datasets described by a dissimilarity, \cite{5}). It also includes implementations of Generative Topographic Mapping \cite{6} and of LVQ and metric learning for LVQ \cite{7}, \cite{8}. The toolbox is fully documented and illustrated in the book \cite{2}.

Since the free statistical software \texttt{R} \cite{9}, is one of the most (if not the most) widely used program in the statistical and bioinformatics communities, we have implemented a full \texttt{R} package providing various versions of the SOM algorithm and many functions for diagnosis and interpretation. SOMbrero is inspired by Patrick Letremy’s SAS programs\textsuperscript{2}. Prior SOMbrero\textsuperscript{3}, several \texttt{R} packages were proposing SOM implementations (e.g., \texttt{class}, som, popsom, kohonen, ... see \cite{10} for a more precise description of these packages). However, as far as we can tell, SOMbrero is the most complete one, with user-friendly features and implementations of variants of the original algorithm for relational data and contingency tables, based on the stochastic version of the method. Current version of the package is 1.2 (2016/09/02) and the package runs on \texttt{R} (\( \geq 3.0.1 \)) with dependencies to the packages \texttt{wordcloud}, \texttt{scatterplot3d}, \texttt{RColorBrewer}, \texttt{shiny}, \texttt{grDevices}, \texttt{graphics} and \texttt{stats}.

The next sections describe the different features of the package. More precisely Section II presents the general organization of the package and its standard use for numeric datasets. Section III presents applications of SOM to explore contingency tables and Section IV describes a variant for dissimilarity datasets.

II. SOM FOR NUMERIC DATASETS AND GENERAL ORGANIZATION OF SOMbrero

This section provides an overview of the package main features. It fully describes functions that can be used for the standard SOM (that processes numeric datasets). However, most of what is described in this section is not restricted to numeric datasets and also exist for the other two variants of the algorithm that are described in Sections III and IV.

A. Training stochastic SOM for numeric datasets

SOMbrero implements the stochastic (also called “online”) version of the SOM algorithm. More precisely, the dataset \( \mathbf{X} = (x_{ij})_{i=1,...,n,j=1,...,d} \) of \( n \) observations taking values in \( \mathbb{R}^d \), is mapped into a low dimensional grid composed of \( U \) units. Conversely, every unit \( u \) is associated with a prototype \( p_u \in \mathbb{R}^d \). The grid induces a natural distance \( d \) on the map which provides a measure of dissimilarity

\[ d(x, y) = \sum_{d=1}^{d} \frac{|x_d - y_d|^2}{s_d^2} \]

\textsuperscript{1}current version 2.1, last update from December 2012, available at http://research.ics.aalto.fi/software/somtoolbox/

\textsuperscript{2}current version 9.1.3, last updated in December 2005, no longer maintained, available at http://samos.univ-paris1.fr/Programmes-bases-sur-l-algorithme

\textsuperscript{3}https://CRAN.R-project.org/package=SOMbrero
between every pair of units, \(d(u, u')\). The stochastic version of the algorithm processes the observations one by one, with the iterative application of two steps:

- an assignment step where one observation (on-line version) is classified into the unit with the closest prototypes (according to \(\mathbb{R}^d\) Euclidean distance);
- a representation step where all prototypes are updated according to the new assignment. For the on-line version of the algorithm, this step is performed by mimicking a stochastic gradient descent scheme.

The method is fully described in Algorithm 1, in which \(H^t\) is the neighborhood function that satisfies \(H^t : \mathbb{R}^+ \to \mathbb{R}^+\), \(H^t(0) = 1\) and \(\lim_{t \to +\infty} H^t(z) = 0\), and \(\mu(t)\) is a training parameter. Generally, \(H^t\) and \(\mu(t)\) are supposed to be decreasing with the iteration number, \(t\).

**Algorithm 1** Stochastic SOM algorithm for numeric datasets

**Require:** Numeric dataset: \(X = (x_{ij})_{i=1,...,n,j=1,...,d}\)
1: For all \(u = 1,...,U\) initialize \(p_{u0}^t\) randomly in \(\mathbb{R}^d\)
2: for \(t = 1,...,T\) do
3: Randomly choose an input \(i \in \{1,...,n\}\)
4: Assignment step: find the unit of the prototype closest to \(x_i = (x_{i1},...,x_{id})\):
   \[
   f^t(x_i) \leftarrow \arg \min_{u=1,...,U} \|x_i - p_{u}^{t-1}\|^2
   \]
5: Representation step: \(\forall u = 1,...,U,\)
   \[
   p_u^t \leftarrow p_u^{t-1} + \mu(t) H^t(d(f^t(x_i), u))(x_i - p_{u}^{t-1})
   \]
   where \(1_i\) is a vector with a single non null coefficient at the \(i\)th position, equal to one.
6: end for
7: return Clustering: \((f^T(x_i))_{i=1,...,n}\) and prototypes: \((p_u^T)_{u=1,...,U}\).

In **SOMbrero**, the following options can be passed to the function `trainSOM` that trains a SOM:

- the grid of the SOM is a square grid with \(U = m_1 \times m_2\) units \((m_1\) and \(m_2\) chosen by the user). Units are positioned in \(\mathbb{R}^2\) at \((k_1(u), k_2(u))_{k_1(u)=1,...,m_1,k_2(u)=1,...,m_2}\). Default values are approximately \(m_1 = m_2 = \sqrt{n}\) (with a minimum of 5 and a maximum of 10 for default values);
- the distance between units on the grid, \(d\), is Euclidean (default) or any type of distance available in the \(\mathbb{R}\) function `dist` or also a relationship denoted by `letremy` which corresponds to the original implementation by Patrick Letremy and switches between Euclidean distance and the distance “maximum” which is the maximum between the distance on the first and the second coordinates;
- the function \(H^t\), also called neighborhood relationship, can be chosen as Gaussian (default), \(H^t(z) = e^{-z^2/r(t)}\) in which \(r(t)\) is decreasing with \(t\), or can be of type `letremy`: \(H^t(z) = \begin{cases} 0 & \text{if } z \leq r(t) \\ 1 & \text{otherwise} \end{cases}\), with \(r(t)\) decreasing during the training.
- prototypes can be initialized randomly as described in step 1 of Algorithm 1 or to one observation of the dataset, randomly chosen, each or by positioning them regularly on the first to PC of a PCA performed on \(X\) or finally, to values specified by the user;
- \(X\) can be preprocessed by centering and scaling to unit variance or not preprocessed at all. In the case of a preprocessing of the data, the prototypes are returned in the original scale of \(X\);
- the assignment step can be performed in a standard way, as described in step 4 of Algorithm 1, or using the Heskes’s modified assignment step [11]:
   \[
   f^t(x_i) \leftarrow \arg \min_{u=1,...,U} \sum_{u'=1}^U H^t(d(u, u'))\|x_i - p_{u'}^{t-1}\|^2;
   \]
- the number of iterations \(T\) can be set by the user. Default values is equal to 5n;
- finally, the user can chose to save intermediate states of the training, which includes saving the prototypes, the clustering for all observations and the value of
   \[
   E^t = \sum_{i=1}^n \sum_{u=1}^U H^t(d(f^t(x_i), u))\|x_i - p_{u}^{t}\|^2.
   \]

**B. Plots and diagnosis functions**

**SOMbrero** has been conceived to help the user interpret the output of the algorithm. More precisely, two standard quality measures [12] are given with the function `quality` that computes:

- the topographic error, which is the average frequency (over all observations) on which the prototype that comes second closest to an observation is not in the direct neighborhood (on the grid) of the winner prototype. It is a real number between 0 and 1, a value close to 0 indicating good quality;
- the quantization error, computed as:
  \[
  Q = \frac{1}{n} \sum_{i=1}^n \|x_i - p_{f(x_i)}\|^2.
  \]

Moreover, **SOMbrero** comes with many different plots, that can be used to represent the original dataset, the prototypes or an additional variable related to the observations, on the grid. 16 types of plots are available with the single function `plot` and two options `what` (what to plot? observations, prototypes or additional information) and `type` (which type of graph to plot?). Some of these plots display distances between prototypes on the grid and can be used as diagnosis plots. An exhaustive description of available graphics is provided in [10] and examples of such plots, obtained on the famous iris dataset [13], are given in Figure 1.

Finally, super-clusters can be obtained by using the function `superClass` on the output of the `trainSOM` function. This method performs hierarchical clustering on the prototypes of the map to cluster the units in “super-clusters”. These super-clusters can also been visualized on the map in various ways.
Fig. 1: Different types of graphics obtained for the iris dataset: distribution of the variables within units (1a), prototype values for the different dimension (1b), additional variable (Species) distribution within units (1c) and distances between prototypes (1d) as proposed in [14].
C. Documentation and graphical user interface

Additionally to the standard \texttt{R} documentation, \texttt{SOMbrero} also provides five vignettes (documentation files accessible from within the package and at https://cran.r-project.org/web/packages/SOMbrero/). The vignettes illustrate the functions with three datasets that correspond to the three variants implemented in the package. The first dataset is the iris dataset [13]. The second dataset (KORRESP variant, see Section III) is the contingency table of votes for the combinations of “Départements” and “Candidates” during the French presidential election of 2002\(^4\). It provided in the package under the name \texttt{presidentielles2002}. The last dataset (relational variant, see Section IV) is the shortest path length between all nodes in the co-occurrence network obtained from the French novel “Les misérables” (V. Hugo) [15]. This dataset (co-occurrence network and dissimilarity matrix) is provided in the package under the name \texttt{lesmis}.

For users who are not familiar with \texttt{R} command lines, \texttt{SOMbrero} contains a graphical user interface which has been programmed with the \texttt{R} package \texttt{shiny} [16] (see Figure 2). The GUI is launched from within \texttt{R} with the function \texttt{sombreroGUI()} and opens in default web browser.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{sombrero_gui.png}
\caption{Screenshot of the SOMbrero web user interface.}
\end{figure}

III. SOM FOR CONTINGENCY TABLES

Another very standard type of data that can be handled by \texttt{SOMbrero} is the case of contingency tables in which the dataset \( T = (n_{ij})_{i=1,\ldots,p; j=1,\ldots,q} \) is composed of joint frequencies for a pair of categorical variables (\( p \) is the number of levels for the first categorical variable and \( q \) the number of levels for the second). Classical correspondence analysis performs a weighted principal components analysis, using the \( \chi^2 \) distance simultaneously on the row profiles and on the column profiles. The same principle is used in the so-called “KORRESP” algorithm, [17], [18], which extends SOM to contingency tables and is implemented in \texttt{SOMbrero}.

More precisely, the contingency table is first transformed into a dataset with row and column profiles:

\[
X = \begin{pmatrix}
\text{columns} & \text{rows}
\end{pmatrix}
\begin{pmatrix}
\text{row profile} & \text{augmented row profile} & \text{rows}
\end{pmatrix}
\begin{pmatrix}
\text{augmented column profile} & \text{column profile} & \text{columns}
\end{pmatrix}
\]

for which:

- row profiles, \((x_i)_{i=1,\ldots,p}\) are defined by \( \forall j = 1,\ldots,q, x_{ij} = \frac{n_{ij}}{n_j} \) and column profiles, \((x_i)_{i=p+1,\ldots,p+q}\) are defined similarly by \( \forall j = q+1,\ldots,q+p, x_{ij} = \frac{n_{ij}}{n_{i,p}} \).
- augmented row profiles correspond to the most probable column profile for that row: \((\tilde{x}_i)_{i=1,\ldots,p}\) are defined by \( \forall j = q+1,\ldots,q+p, \tilde{x}_{ij} = x_{k(i)+p,j} \) with \( k(i) = \arg \max_{k=1,\ldots,q} x_{ik} \). Similarly, augmented column profiles are \((\tilde{x}_i)_{i=p+1,\ldots,p+q}\) with \( \forall j = 1,\ldots,q, \tilde{x}_{ij} = x_{k(i)-q,j} \) and \( k(i) = \arg \max_{k=q+1,\ldots,q+p} x_{ik} \).

The algorithm is fully described in Algorithm 2. Prototypes are defined in \( \mathbb{R}^{p+q} \). The algorithm then alternatively processes a row and a column (randomly chosen during the stochastic training process). For the chosen row (or column),

- the assignment step uses the profile \((x_i)_{i=1,\ldots,p}\) \( \text{in} \mathbb{R}^q \) \( \text{for rows} \) or \((x_i)_{i=p+1,\ldots,p+q}\) \( \text{in} \mathbb{R}^p \) \( \text{for columns} \). For a row \( i \in \{1,\ldots,p\} \), this gives:

\[
\begin{align*}
&f^t(x_i) \leftarrow \arg \min_{u=1,\ldots,U} \| x_i - (p^t_{uj})_{j=1,\ldots,q} \|^2; \\
&\text{the representation step uses the augmented profile (in} \mathbb{R}^{p+q}\text{). For the chosen row} i, \text{this gives:}
\end{align*}
\]

\[
p^t_u \leftarrow p^t_{u-1} + \mu(t) H^t(d(f^t(x_i), u)) \left( \begin{bmatrix} x_i \\ \tilde{x}_i \end{bmatrix} - p^t_{u-1} \right).
\]

This method can be used in \texttt{SOMbrero} with \texttt{trainSOM(..., type= "korresp")} and provides a map in which levels for both variables of the contingency tables are clustered simultaneously. This approach is analogous to the common representation provided for contingency tables in correspondence analysis.

IV. SOM FOR RELATIONAL DATASETS

A. Relational SOM

In the case where the observations \((x_i)_{i=1,\ldots,n}\) take values in an arbitrary input space \( G \), a widely used representation of the data is to compute a measures of dissemblance or resemblance between pairs of observations. Well known examples of such frameworks include shortest path lengths between pairs of nodes in a graph [19], string edit distance between categorical time series [20] or DNA samples [21] or various kernels used in many application fields, including biology [22].

In this section, we will consider that the data are described by
Algorithm 2 KORRESP variant of the stochastic SOM algorithm for contingency tables

Require: Contingency table: \( T = (n_{ij})_{i=1,\ldots,p; j=1,\ldots,q} \)

1: Define \( X \) with row/column profiles and augmented row/column profiles
2: For all \( u = 1, \ldots, U \) initialize \( p_{u}^{0} \) randomly in \( \mathbb{R}^{p+q} \)
3: for \( t = 1, \ldots, T \) do
4: \hspace{1em} Start Process rows
5: \hspace{2em} Randomly choose an input \( i \in \{1, \ldots, p\} \)
6: \hspace{2em} Assignment step: find the unit of the prototype closest to \( x_{i} \):
\[
f^{t}(x_{i}) \leftarrow \arg \min_{u=1, \ldots, U} \| x_{i} - (p_{u}^{t-1})_{j=1,\ldots,q} \|^2
\]
7: \hspace{2em} Representation step: \( \forall u = 1, \ldots, U, \)
\[
p_{u}^{t} \leftarrow p_{u}^{t-1} + \mu(t)H^{t}(d(f^{t}(x_{i}), u)) \left( \begin{bmatrix} x_{i} \\ \beta_{u} \end{bmatrix} - p_{u}^{t-1} \right)
\]
where \( 1_{i} \) is a vector with a single non null coefficient at the \( i \)th position, equal to one.
8: \hspace{1em} End
9: \hspace{1em} Start Process columns
10: \hspace{2em} Randomly choose an input \( i \in \{p+1, \ldots, p+q\} \)
11: \hspace{2em} Assignment step: find the unit of the prototype closest to \( x_{i} \):
\[
f^{t}(x_{i}) \leftarrow \arg \min_{u=1, \ldots, U} \| x_{i} - (p_{u}^{t-1})_{j=q+1,\ldots,q+p} \|^2
\]
12: \hspace{2em} Representation step: \( \forall u = 1, \ldots, U, \)
\[
p_{u}^{t} \leftarrow p_{u}^{t-1} + \mu(t)H^{t}(d(f^{t}(x_{i}), u)) \left( \begin{bmatrix} \beta_{u} \\ x_{i} \end{bmatrix} - p_{u}^{t-1} \right)
\]
where \( 1_{i} \) is a vector with a single non null coefficient at the \( i \)th position, equal to one.
13: \hspace{1em} End
14: \hspace{1em} end for
15: return Clustering: \( (f^{T}(x_{i}))_{i=1,\ldots,p+q} \) and prototypes: \( (p_{u}^{T})_{u=1,\ldots,U} \).

Algorithm 3 Relational variant of the stochastic SOM algorithm for dissimilarity data

1: For all \( u = 1, \ldots, U \) and \( i = 1, \ldots, n \), initialize \( \beta_{u}^{0} \) such that \( \beta_{u}^{0} \geq 0 \) and \( \sum_{u=1}^{n} \beta_{u}^{0} = 1 \).
2: for \( t = 1, \ldots, T \) do
3: \hspace{1em} Randomly choose an input \( x_{i} \)
4: \hspace{1em} Assignment step: find the unit of the closest prototype
\[
f^{t}(x_{i}) \leftarrow \arg \min_{u=1, \ldots, U} \left( \beta_{u}^{t-1} \Delta \right)_{i} - \frac{1}{2} \left( \beta_{u}^{t-1} \Delta \beta_{u}^{t-1} \right)
\]
where \( (\beta_{u}^{t-1} \Delta)_{i} \) is the \( i \)th entry of \( \beta_{u}^{t-1} \Delta \)
5: \hspace{1em} Representation step: \( \forall u = 1, \ldots, U, \)
\[
\beta_{u}^{t} \leftarrow \beta_{u}^{t-1} + \mu(t)H^{t}(d(f^{t}(x_{i}), u)) (1_{i} - \beta_{u}^{t-1})
\]
where \( 1_{i} \) is a vector with a single non null coefficient at the \( i \)th position, equal to one.
6: \hspace{1em} end for

If the dissimilarity matrix is a Euclidean distance, then the relational SOM is exactly identical to the standard numerical SOM as long as the prototypes of the original SOM are initialized in the convex hull of the original data (i.e., the initial prototypes can be written \( p_{u}^{0} = \sum_{i} \beta_{u}^{0} x_{i} \)). Similarly, the relational SOM is identical to kernel SOM as described in [27], [30], [31] for a dissimilarity defined from a kernel \( K \) by
\[
\delta(x_{i}, x_{j}) := K(x_{i}, x_{i}) + K(x_{j}, x_{j}) - 2K(x_{i}, x_{j}).
\]

B. Acceleration of relational SOM

However, the complexity of the assignment and representation steps of stochastic relational SOM are, respectively, \( O(n^{2}U) \) and \( O(nU) \), as pointed in [32]. This leads to a total complexity of \( O(n^{2}U) \) for one iteration. To obtain good convergence properties, the algorithm requires a number of iterations of the order of \( an \), as shown in [33], yielding a complexity of \( O(an^{3}U) \). Hence, relational SOM is not adapted to large datasets and cannot be used to analyze more than a few thousands observations. [34] have proposed two approximate versions to overcome this issue, using sparse representations of the prototypes or DR pre-processing techniques. However, the version implemented in SOMbrero uses the solution proposed in [35] which yields to a complexity of \( O(an^{2}U) \).

More precisely, the method is based on a re-formulation of the assignment step (step 4 in Algorithm 3):
\[
f^{t}(x_{i}) = \arg \min_{u=1, \ldots, U} B_{u}^{t-1} - \frac{1}{2} A_{u}^{t-1}
\]
in which
\[
A_{u}^{t} = \left( \sum_{j,j'=1}^{n} \beta_{u}^{t}eta_{u}^{j'} \delta_{j,j'} \right)_{u=1,\ldots,U}
\]
and
\[
B_{u}^{t} = \frac{1}{n} \sum_{i} \beta_{u}^{t} x_{i} x_{i}^{T} - \frac{1}{n} \sum_{i} \beta_{u}^{t} \beta_{u}^{j'} \delta_{j,j'} x_{i} x_{i}^{T}
\]
for all \( u = 1, \ldots, U \).
is a vector of size $U$ and

$$B^t = \left( \sum_{j=1}^{n} \beta_{uj}^t \delta_{ij} \right)_{u=1,...,U, i'=1,...,n}$$

is a $(U \times n)$-matrix.

The updates of $A^t$ and $B^t$ are performed during the representation step, which is thus equivalent to $\forall u = 1, \ldots, U, \quad \beta_{uj}^t = (1-\lambda_u(t))\beta_{uj}^{t-1} + \lambda_u(t)1$, in which $\lambda_u(t) = \mu(t)H^t(d(f^t(x_i), u))$. This leads to the following updates:

$$B_{ui'}^t = \sum_{j=1}^{n} \beta_{uj}^t \delta_{i'j} = (1 - \lambda_u(t))B_{ui'}^{t-1} + \lambda_u(t)\delta_{i'j},$$

and

$$A_u^t = \sum_{j,j'=1}^{n} \beta_{uj}^t \beta_{uj'}^t \delta_{jj'}$$

$$= (1 - \lambda_u(t))^2 A_u^{t-1} + \lambda_u(t)^2 \delta_{ii}$$

$$+ 2\lambda_u(t)(1 - \lambda_u(t))B_{ui'}^{t-1}.$$
