Une adaptation des cartes auto-organisatrices pour des données décrites par un tableau de dissimilarités

Aïcha El Golli, Fabrice Rossi, Brieuc Conan-Guez, Yves Lechevallier

Une adaptation des cartes auto-organisatrices pour des données décrites par un tableau de dissimilarités

Aïcha El Golli [a,*], Fabrice Rossi [a],
Brieuc Conan-Guez [b,a], et Yves Lechevallier [a]

[a] *Projet AxIS, INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France*

[b] *LITA EA3097, Université de Metz, Ile du Saulcy, F-57045 Metz, France*

Résumé

De nombreuses méthodes d'analyse des données ne sont applicables qu'aux données qui peuvent être représentées par un nombre fixé de valeurs numériques, alors que la plupart des observations issues de problèmes réels ne se trouvent pas naturellement sous cette forme. Il est alors indispensable d'adapter les méthodes prévues pour le cas vectoriel à des données plus complexes. Une solution très souple et très générale consiste à construire une mesure de (dis)similarité, basée sur le savoir des experts du domaine concerné, qui permette de comparer deux à deux les données étudiées. On construit alors des méthodes d'analyse qui ne travaillent qu'à partir du tableau de dissimilarités résumant les données.

Dans cet article, nous proposons une adaptation des cartes auto-organisatrices de Kohonen à tout type de données pour lesquelles une mesure de (dis)similarité est définie. L'algorithme proposé est une adaptation de la version *batch* de la méthode employée pour les données classiques.

Nous validons notre méthode sur une application réelle, l'analyse de l'usage du site web de l'Institut National de Recherche en Informatique et Automatique à partir de fichiers log de ses serveurs.

Mots clés : Classification non supervisée, Projection non linéaire, Cartes auto-organisatrices, Dissimilarités, Web Usage Mining

Abstract

Many data analysis methods cannot be applied to data that are not represented by a fixed number of real values, whereas most of real world observations are not readily available in such a format. Vector based data analysis methods have therefore to be adapted in order to be used with non standard complex data. A flexible and general solution for this adaptation is to use a (dis)similarity measure. Indeed, thanks to expert knowledge on the studied data, it is generally possible to define a measure that can be used to make pairwise comparison between observations. General data analysis methods are then obtained by adapting existing methods to (dis)similarity matrices.

In this article, we propose an adaptation of Kohonen's Self Organizing Map (SOM) to (dis)similarity data. The proposed algorithm is an adapted version of the vector based batch SOM.

The method is validated on real world data: we provide an analysis of the usage patterns of the web site of the Institut National de Recherche en Informatique et Automatique, constructed thanks to web log mining method.

Key words: Clustering, Nonlinear projection, Self Organizing Map, Dissimilarity, Web Usage Mining

1 Introduction

Dans de nombreuses applications, les observations ne sont pas naturellement représentées sous forme d'un nombre fixé de valeurs numériques, i.e., sous forme de vecteurs de $\mathbb{R}^p$. Les données réelles peuvent en effet être de taille variable, être décrites par des variables qui ne sont pas directement comparables, ne pas être numériques, etc. On peut évoquer par exemple les données textuelles, les données semi-structurées (e.g., les documents XML), les données fonctionnelles (Ramsay and Silverman, 1997) et les données symboliques (intervalles, distributions, etc., cf Bock and Diday 1999). Or, beaucoup de méthodes d'analyse de données ont été construites en exploitant les

* Auteur à contacter.
Adresses électroniques : Aicha.ElGolli@inria.fr (Aïcha El Golli),
Fabrice.Rossi@inria.fr (Fabrice Rossi),
Brieuc.Conan-Guez@iut.univ-metz.fr (Brieuc Conan-Guez),
Yves.Lechevallier@inria.fr (Yves Lechevallier).
0 Article à paraître dans la Revue de Statistique Appliquée

propriétés de $\mathbb{R}^p$, plus particulièrement les opérations classiques qu'on peut appliquer aux vecteurs de cet espace : combinaisons linéaires, produit scalaire et norme. Pour être appliquées à des données non vectorielles, les méthodes en question doivent être modifiées et adaptées. Dans certaines conditions, notamment pour les données symboliques, il est possible de mettre en œuvre des techniques de représentations numériques (De Reyniès, 2002, 2003; Chavent and Lechevallier, 2002; Chavent et al., 2003; Verde et al., 2000) pour se ramener au cas vectoriel. De la même façon, l'utilisation d'un opérateur de projection permet d'associer à des données fonctionnelles une représentation vectorielle satisfaisante (Ramsay and Silverman, 1997).

Ces approches ont cependant une portée limitée, en particulier car la représentation numérique induit souvent une perte d'information. Une alternative particulièrement fructueuse consiste à s'appuyer sur la définition de mesures de (dis)similarités entre données complexes et à généraliser les méthodes classiques d'analyse de données au cas de tableaux de (dis)similarités. L'avantage évident de cette stratégie est de séparer la construction d'algorithmes d'analyse du choix de la représentation des données. Cela permet de proposer une implémentation unique d'un algorithme d'analyse qui pourra être utilisée avec toute sorte de données, à condition de pouvoir calculer une (dis)similarités entre les observations. L'algorithme et son implémentation deviennent alors universels. De plus, il devient possible de faire appel à des experts pour définir la (dis)similarité utilisée, sans que ceux-ci n'aient besoin de connaître l'algorithme utilisé : seule la pertinence de la comparaison entre les données est importante pour les experts. On peut ainsi réutiliser les nombreux travaux réalisés dans le domaine, comme par exemple Bock and Diday (1999, chapitre 8 *"Similarity and dissimilarity"*, pages 139-197) pour les données symboliques et Wang et al. (1999) pour les données semi-structurées.

Nous proposons dans le présent article une adaptation de l'algorithme des cartes auto-organisatrices (Kohonen, 1995, 1997 & 2001) au cas de données décrites par un tableau de dissimilarités. Les cartes auto-organisatrices sont un instrument très utile pour l'analyse exploratoire des données car elles combinent une classification avec une projection non linéaire. Le principe fondamental de cette méthode est de représenter un ensemble d'observations grâce à des prototypes (aussi appelés référents) organisés selon une structure fixée *a priori*. Les prototypes doivent réaliser une bonne quantification des données d'origine : chaque observation est affectée à un prototype, ce qui définit une partition des données. Chaque prototype est alors représentatif de la classe des observations qui lui sont affectées, ce qui donne aux cartes auto-organisatrices des aspects similaires à l'algorithme des *k-means* (MacQueen, 1965). De plus, chaque prototype tient compte des observations affectées aux classes voisines. Enfin, la structure *a priori* permet une représentation graphique des prototypes, en général sur un plan. Cette représentation s'apparente à une projec-

tion non linéaire, car la cohérence des classes et la représentativité des prototypes autorisent à considérer ces derniers comme les projetés des données d'origine. Les cartes auto-organisatrices fournissent ainsi une alternative à des méthodes classiques de projections, linéaires comme les méthodes factorielles (Hotelling, 1933) ou non linéaires comme le *Multi Dimensional Scaling* (Torgerson, 1952) ou Isomap (Tenenbaum et al., 2000).

La généralisation que nous proposons s'adapte à toutes données pour lesquelles une dissimilarité peut être définie. Elle est donc plus générale que les extensions spécifiques des cartes auto-organisatrices qui ont été proposées pour certaines données complexes comme les données symboliques (Bock, 2001), les données structurées de type séries temporelles, arbres ou graphes (Hammer et al., 2004), les chaînes de caractères (Somervuo, 2004), les données qualitatives (Cottrell et al., 2004; Cottrell and Letrémy, 2005) et les données fonctionnelles (Rossi et al., 2004).

Dans la section 2, nous commencerons par rappeler l'algorithme *batch* des cartes auto-organisatrices. Nous montrerons dans la section 3 comment l'algorithme peut être adapté au cas de données décrites uniquement par un tableau de dissimilarités. Nous conclurons cet article par une application de notre algorithme à un problème réel d'analyse de l'usage d'un site Web, présenté dans la section 4. Nous verrons à cette occasion que l'algorithme proposé donne des résultats très satisfaisants en terme d'analyse du site web de l'Institut National de Recherche en Informatique et Automatique (INRIA).

2 Les cartes auto-organisatrices

2.1 Principe général

L'algorithme des cartes auto-organisatrices de Kohonen (Kohonen, 1995, 1997 & 2001), abrégé en SOM pour *Self Organizing Map*, est à la fois un algorithme de projection non linéaire et un algorithme de classification. Il associe à des données d'origine (appartenant en général à un espace de grande dimension) un ensemble de prototypes organisés selon une structure de faible dimension (en général deux) choisie *a priori*. Chaque prototype représente un sous-ensemble des données d'origine qu'on peut considérer comme une classe. L'organisation des prototypes, et donc des classes, est imposée par la structure *a priori*, mais elle est aussi contrainte par les données elles-mêmes de sorte que la représentation graphique des prototypes réalise une projection non linéaire des données qui préserve leur topologie.

Plus formellement, la structure (c'est-à-dire la *carte*) est décrite par un graphe

(C, Γ). C désigne les M *neurones* de la carte. Chaque neurone est associé à un prototype (aussi appelé référent du neurone) et à une classe (on aura donc M classes). L'organisation *a priori* provient de l'ensemble d'arêtes Γ : deux neurones c et r sont connectés directement et donc voisins dans la carte si $(c, r) \in \Gamma$. Cette structure de graphe induit une distance discrète δ sur la carte : pour tout couple de neurones (c, r) de la carte, la distance $\delta(c, r)$ est définie comme étant la longueur du plus court chemin entre c et r.

Le but de l'algorithme SOM est, partant d'un ensemble de N observations, les $\mathbf{x}_1, \ldots, \mathbf{x}_N$ (qui forment l'ensemble Ω), d'associer à chaque neurone $c \in C$ un prototype $\mathbf{p}_c$ et un sous-ensemble $\mathcal{C}_c$ de Ω. On demande que les $(\mathcal{C}_c)_{c \in C}$ forment une partition de Ω et que pour tout c, $\mathbf{p}_c$ représente de façon satisfaisante les éléments de $\mathcal{C}_c$ (il s'agit d'une mesure de qualité de la partition) : ceci correspond à l'aspect classificatoire de l'algorithme SOM. De plus il faut que la structure *a priori* soit respectée, c'est-à-dire que si c et r sont des neurones proches (au sens de la distance δ induite par le graphe Γ), alors $\mathbf{p}_c$ doit représenter correctement les éléments de $\mathcal{C}_r$ (de même pour $\mathbf{p}_r$ par rapport à $\mathcal{C}_c$).

2.2 Les cartes auto-organisatrices pour les données classiques

Dans la section précédente, nous sommes volontairement restés très vagues dans la description des données, des prototypes et de la notion de proximité (ou de représentation satisfaisante), afin de rappeler le principe général des cartes auto-organisatrices. Nous allons maintenant rappeler en détail la version dite *batch* de l'algorithme pour des données classiques, c'est-à-dire des éléments d'un espace vectoriel $\mathbb{R}^p$. Dans cette situation, Ω est donc un ensemble de N vecteurs de $\mathbb{R}^p$. Les prototypes (les $\mathbf{p}_c$) sont aussi choisis dans $\mathbb{R}^p$. Enfin, les données sont comparées au sens de la distance euclidienne.

Pour formaliser la notion de respect de la structure *a priori* et de qualité de la partition, on utilise une fonction noyau K de $\mathbb{R}^+$ dans $\mathbb{R}^+$, décroissante et telle que $K(0) = 1$ et $\lim_{x \to \infty} K(x) = 0$ (en pratique, on utilise souvent $K(x) = e^{-x^2}$). Cette fonction engendre une famille de fonctions, les K^T, définies par $K^T(x) = K(\frac{x}{T})$. Le paramètre T est analogue à une température (Thiria et al., 1997; Dreyfus et al., 2002) : quand T est élevé, $K^T(x)$ reste proche de 1 même pour de grandes valeurs de x ; au contraire une valeur faible engendre une fonction K^T qui décroît très vite vers 0. Le rôle de K^T est de transformer la distance discrète δ induite par la structure de graphe en une fonction de voisinage plus régulière et paramétrée par T : on utilisera ainsi $K^T(\delta(c, r))$ comme mesure de proximité effective entre les neurones c et r. Pendant le déroulement de l'algorithme SOM, la valeur de T décroît afin d'assurer la stabilisation de la solution obtenue.

La qualité de la partition $(\mathcal{C}_c)_{c\in C}$ et des prototypes associés, les $(\mathbf{p}_c)_{c\in C}$, est alors donnée par l'énergie suivante (Cheng, 1997), qui doit être la plus faible possible :

$$E^T\left((\mathcal{C}_c)_{c\in C},(\mathbf{p}_c)_{c\in C}\right)=\sum_{\mathbf{x}_i\in\Omega}\sum_{c\in C}K^T\left(\delta(f(\mathbf{x}_i),c)\right)\left\|\mathbf{p}_c-\mathbf{x}_i\right\|^2, \tag{1}$$

où f désigne la fonction d'affectation, telle que $f(\mathbf{x}_i)=c$ si $\mathbf{x}_i\in\mathcal{C}_c$. Pour simplifier la suite du texte, on note $\mathcal{P}=(\mathcal{C}_c)_{c\in C}$ la partition et $\mathcal{R}=(\mathbf{p}_c)_{c\in C}$ le système de prototypes associé.

Pour bien comprendre le sens de l'énergie, on peut la récrire de la façon suivante :

$$E^T\left(\mathcal{P},\mathcal{R}\right)=\sum_{r\in C}\sum_{\mathbf{x}_i\in\mathcal{C}_r}\sum_{c\in C}K^T\left(\delta(r,c)\right)\left\|\mathbf{p}_c-\mathbf{x}_i\right\|^2. \tag{2}$$

Comme $K^T(0)=1$, on peut décomposer l'énergie en deux termes :

$$E^T\left(\mathcal{P},\mathcal{R}\right)_R=\sum_{r\in C}\sum_{\mathbf{x}_i\in\mathcal{C}_r}\left\|\mathbf{p}_r-\mathbf{x}_i\right\|^2, \tag{3}$$

et

$$E^T\left(\mathcal{P},\mathcal{R}\right)_S=\sum_{r\in C}\sum_{c\neq r}\sum_{\mathbf{x}_i\in\mathcal{C}_c}K^T\left(\delta(r,c)\right)\left\|\mathbf{p}_r-\mathbf{x}_i\right\|^2. \tag{4}$$

Le terme $E^T\left(\mathcal{P},\mathcal{R}\right)_R$ correspond à la distorsion utilisée dans les algorithmes de classification de type *k-means* (MacQueen, 1965). Le terme $E^T\left(\mathcal{P},\mathcal{R}\right)_S$ est lui spécifique aux cartes auto-organisatrices. On voit qu'il impose au prototype du neurone r de représenter les observations qui ont été affectées à d'autres neurones. Un défaut de représentation, c'est-à-dire une grande valeur pour $\left\|\mathbf{p}_r-\mathbf{x}_i\right\|^2$, pèse d'autant plus lourdement dans l'énergie que le neurone r est proche (dans la structure *a priori*) du neurone $f(\mathbf{x}_i)$.

La minimisation de l'énergie $E^T\left(\mathcal{P},\mathcal{R}\right)$ est un problème d'optimisation combinatoire. On se contente en pratique d'une solution sous-optimale obtenue par une heuristique. Une telle heuristique est donnée par la version *batch* de l'algorithme SOM (Kohonen, 1995, 1997 & 2001) et par ses variantes (Heskes and Kappen, 1993; Cheng, 1997; Thiria et al., 1997). Les variantes de l'algorithme alternent deux étapes distinctes, une étape d'affectation (calcul de f) et une étape de représentation (calcul des $\mathbf{p}_c$), ce qui le classe dans les algorithmes de type nuées dynamiques (Diday, 1971; Diday and Simon, 1976; Diday and Govaert, 1977).

L'algorithme *batch* initial (Kohonen, 1995, 1997 & 2001) utilise une étape d'affectation de type *winner takes all* définie par

$$f(\mathbf{x})=\arg\min_{r\in C}\left\|\mathbf{p}_r-\mathbf{x}_i\right\|^2. \tag{5}$$

La convergence de cette version de l'algorithme a été étudiée dans (Fort et al.,

2001), mais les résultats obtenus sont difficilement extensibles aux cas des données décrites par des dissimilarités.

Pour simplifier l'analyse, nous utilisons dans le présent article la variante proposée par (Heskes and Kappen, 1993). L'étape d'affectation consiste ici à minimiser $E^T(\mathcal{P}, \mathcal{R})$ en considérant les prototypes fixés. Comme dans l'algorithme *batch* standard, l'étape de représentation minimise la même énergie mais en considérant les classes comme fixées. Bien que les deux optimisations soient réalisées de façon exacte, on ne peut pas garantir que l'énergie est globalement minimisée par cet algorithme. Par contre, si on fixe la structure de voisinage (pour T fixé), l'algorithme converge en un nombre fini d'étape vers un état stable (Cheng, 1997).

Comme l'énergie est une somme de termes indépendants, on peut remplacer les deux problèmes d'optimisation par un ensemble de problèmes simples équivalents. La formulation de l'équation 1 montre que l'énergie est construite comme la somme sur l'ensemble des observations d'une mesure d'adéquation de $\mathbb{R}^p \times C$ dans $\mathbb{R}^+$ définie par :

$$\gamma^T(\mathbf{x}, r) = \sum_{c \in C} K^T\left(\delta(r, c)\right) \left\|\mathbf{p}_c - \mathbf{x}\right\|^2, \tag{6}$$

ce qui donne

$$E^T(\mathcal{P}, \mathcal{R}) = \sum_{\mathbf{x}_i \in \Omega} \gamma^T(\mathbf{x}_i, f(\mathbf{x}_i)). \tag{7}$$

Pour optimiser E^T en gardant les prototypes fixés, il suffit donc de minimiser chacune des sommes indépendamment, ce qui amène à définir f comme suit :

$$f(\mathbf{x}) = \arg\min_{r \in C} \gamma^T(\mathbf{x}, r). \tag{8}$$

De même, quand les classes sont fixées, l'optimisation de E^T par rapport aux prototypes s'obtient en minimisant l'énergie associée à chaque neurone, à savoir :

$$E_c^T(\mathbf{p}) = \sum_{\mathbf{x}_i \in \Omega} K^T\left(\delta(f(\mathbf{x}_i), c)\right) \left\|\mathbf{p} - \mathbf{x}_i\right\|^2, \tag{9}$$

ce qui revient à poser :

$$\mathbf{p}_c = \arg\min_{\mathbf{p} \in \mathbb{R}^p} E_c^T(\mathbf{p}). \tag{10}$$

Ce problème d'optimisation admet une solution simple définie comme la moyenne pondérée des observations :

$$\mathbf{p}_c = \frac{\sum_{\mathbf{x}_i \in \Omega} K^T\left(\delta(f(\mathbf{x}_i), c)\right) \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \Omega} K^T\left(\delta(f(\mathbf{x}_i), c)\right)} \tag{11}$$

La version *batch* de l'algorithme SOM étudiée est alors celle décrite dans l'algorithme 1.

Algorithme 1 La version *batch* de l'algorithme SOM

1: Choisir une valeur initiale pour les prototypes $(\mathbf{p}_c)_{c\in C}$ {Étape d'initialisation}

2: **Pour** $l = 1$ à L **faire**

3: **Pour tout** élément $\mathbf{x}$ de Ω **faire** {Étape d'affectation}

4: calculer

$$f(\mathbf{x}) = \arg\min_{r\in C} \gamma^T(\mathbf{x}, r)$$

5: **Fin pour**

6: **Pour tout** neurone $c \in C$ **faire** {Étape de représentation}

7: calculer

$$\mathbf{p}_c = \frac{\sum_{\mathbf{x}_i\in\Omega} K^T\left(\delta(f(\mathbf{x}_i), c)\right)\mathbf{x}_i}{\sum_{\mathbf{x}_i\in\Omega} K^T\left(\delta(f(\mathbf{x}_i), c)\right)}$$

8: **Fin pour**

9: **Fin pour**

Bien que cela n'apparaisse pas explicitement dans l'algorithme, la température T évolue en fonction de l, en général selon une décroissance exponentielle avec l. D'autre part, nous ne développons pas ici les différentes méthodes d'initialisation disponibles pour le choix des valeurs initiales des prototypes : de nombreuses variantes existent (cf Kohonen, 1995, 1997 & 2001; Thiria et al., 1997; Dreyfus et al., 2002).

Enfin, il existe aussi de nombreuses variantes de l'algorithme des cartes auto-organisatrices : nous nous focalisons sur la version *batch* que nous venons de décrire car elle s'adapte simplement au cas d'un tableau de dissimilarités, comme nous allons le voir dans la section suivante. Notons que le caractère déterministe des algorithmes *batch* est central dans les démonstrations de leurs convergences (Cheng, 1997; Fort et al., 2001). L'algorithme stochastique classique est beaucoup plus délicat à analyser (cf par exemple Cottrell et al., 1998).

3 Adaptation à un tableau de dissimilarités

3.1 Une carte et son énergie

Comme nous l'avons indiqué en introduction, notre but est d'adapter les cartes auto-organisatrices au cas de données décrites uniquement par l'intermédiaire d'un tableau de dissimilarités. La différence fondamentale avec les sections

précédentes est que l'ensemble des observations Ω n'est plus une partie de $\mathbb{R}^p$ mais un ensemble quelconque associé à une fonction d, de $\Omega \times \Omega$ dans $\mathbb{R}^+$ qui vérifie les propriétés suivantes :
– d est symétrique, i.e., $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$;
– d est positive, i.e., $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$;
– $d(\mathbf{x}_i, \mathbf{x}_i) = 0$.
La fonction d est donc une dissimilarité : $d(\mathbf{x}_i, \mathbf{x}_j)$ est d'autant plus faible que $\mathbf{x}_i$ et $\mathbf{x}_j$ sont "semblables".

Nous ne faisons aucune hypothèse structurelle sur Ω, ce qui signifie qu'aucune opération n'est possible sur cet ensemble, excepté le calcul de d. Malgré cela, l'étude de l'équation 1 montre que la notion de carte auto-organisatrice et d'énergie associée est généralisable à la situation qui nous intéresse. En effet, l'énergie est définie à partir de la distance euclidienne (au carré) entre les observations et les prototypes. Le reste de l'équation 1 ne fait apparaître que la structure *a priori* et la partition des données. Il est donc tentant de remplacer la distance dans l'équation 1 par la dissimilarité définie sur Ω. Ceci n'est possible que si les prototypes sont contraints à être des éléments de Ω.

Dans certaines situations, cette dernière contrainte peut être considérée comme trop forte car Ω peut être un échantillon peu représentatif de l'espace de départ. Nous proposons donc de généraliser la notion de prototype ou de référent d'un neurone : au lieu d'associer au neurone c un unique prototype $\mathbf{p}_c$ (choisi donc dans Ω), nous lui associons un sous-ensemble A_c contenant q éléments distincts de Ω. Le paramètre q est déterminé en fonction du problème : si la valeur $q = 1$ conduit à des résultats difficiles à interpréter, on peut augmenter q afin de réduire les effets de l'échantillonnage.

L'énergie d'une carte auto-organisatrice ainsi définie est donc :

$$E^T\left((\mathcal{C}_c)_{c\in C}, (A_c)_{c\in C}\right) = \sum_{\mathbf{x}_i\in\Omega}\sum_{c\in C} K^T\left(\delta(f(\mathbf{x}_i), c)\right) \sum_{\mathbf{x}_j\in A_c} d(\mathbf{x}_i, \mathbf{x}_j). \qquad (12)$$

Comme dans le cas classique, on utilisera dans la suite du texte les notations $\mathcal{P} = (\mathcal{C}_c)_{c\in C}$ et $\mathcal{R} = (A_c)_{c\in C}$.

3.2 Interprétation de l'énergie

Cette énergie généralise clairement l'énergie de l'équation 1 au cas d'un tableau de dissimilarités. On peut cependant s'interroger sur sa signification. Comme pour le cas classique, elle se décompose en une partie qui mesure la qualité de la classification et une autre qui impose le respect de la structure *a priori*.

La qualité de la classification est donc définie par :

$$E^T\left(\mathcal{P},\mathcal{R}\right)_R = \sum_{r\in C}\sum_{\mathbf{x}_i\in\mathcal{C}_r}\sum_{\mathbf{x}_j\in A_r} d(\mathbf{x}_i,\mathbf{x}_j), \tag{13}$$

ce qui se réduit à

$$E^T\left(\mathcal{P},\mathcal{R}\right)_R = \sum_{r\in C}\sum_{\mathbf{x}_i\in\mathcal{C}_r} d(\mathbf{x}_i,\mathbf{p}_r), \tag{14}$$

quand $q = 1$, c'est-à-dire le cas le plus simple. Le principe de cette mesure est donc de dire qu'une partition est de bonne qualité si on peut trouver pour chaque classe un prototype (ou un ensemble de prototypes) tels que les éléments de la classe soient semblables au(x) prototype(s) (au sens de la dissimilarité). De la même façon, le respect de la structure impose que le(s) prototype(s) associé(s) à un neurone soi(en)t semblable(s) aux observations associées aux neurones voisins.

Si la dissimilarité correspond à la distance euclidienne au carré, l'énergie retenue est très satisfaisante. En effet, les propriétés de la distance euclidienne (en particulier le théorème de Huygens) font que les classes obtenues en minimisant l'énergie sont compactes et bien séparées : les observations dans une même classe sont proches les unes des autres (compacité) et sont éloignées des observations des autres classes (séparation).

Dans le cas d'une dissimilarité métrique, c'est-à-dire qui vérifie l'inégalité triangulaire (i.e., $d(u,w) \leq d(u,v) + d(v,w)$), la situation est aussi satisfaisante. En effet, deux éléments d'une même classe sont proches car $d(\mathbf{x}_i,\mathbf{x}_j) \leq d(\mathbf{x}_i,\mathbf{p}_c) + d(\mathbf{p}_c,\mathbf{x}_j)$ et que par construction les prototypes sont proches des observations de leur classe.

Par contre, pour une dissimilarité quelconque, les classes peuvent très bien ne pas être compactes. Si l'inégalité triangulaire n'est pas vérifiée, les observations affectées à un neurone peuvent être très proches du ou des prototypes associés au neurone sans être proches entre elles.

En pratique cependant, le but des cartes auto-organisatrices est bien atteint par la minimisation de l'énergie choisie. L'obtention de classes compactes n'est pas en effet le but principal : il s'agit en fait de représenter simplement (en deux dimensions en général) un ensemble de prototypes qui peuvent être considérés comme représentatifs des observations d'origine. L'énergie impose ici que les prototypes soient proches des observations, ce qui correspond bien au but fixé.

3.3 L'algorithme

L'algorithme SOM adapté à un tableau de dissimilarités se définit alors sur le modèle de la version *batch* pour données classiques. On cherche en effet à

minimiser l'énergie (12) de façon heuristique, en alternant une étape d'optimisation par rapport à $\mathcal{P} = (\mathcal{C}_c)_{c\in C}$ (l'affectation) avec une étape d'optimisation par rapport à $\mathcal{R} = (A_c)_{c\in C}$ (la représentation). Comme dans la section précédente, ces optimisations se décomposent en des ensembles d'optimisations simples.

On commence par transposer l'équation (6) au cas des dissimilarités, ce qui donne :

$$\gamma^T(\mathbf{x}, r) = \sum_{c\in C} K^T\left(\delta(r, c)\right) \sum_{\mathbf{x}_j\in A_c} d(\mathbf{x}, \mathbf{x}_j), \tag{15}$$

et

$$E^T\left(\mathcal{P}, \mathcal{R}\right) = \sum_{\mathbf{x}_i\in\Omega} \gamma^T(\mathbf{x}_i, f(\mathbf{x}_i)). \tag{16}$$

Exactement comme dans le cas classique, la phase d'affectation consiste donc à trouver $r = f(\mathbf{x}_i)$ qui minimise $\gamma^T(\mathbf{x}_i, r)$.

Pour la phase de représentation, on définit l'énergie associée au neurone c, sur le modèle de l'équation (9) :

$$E_c^T(A) = \sum_{\mathbf{x}_i\in\Omega} K^T\left(\delta(f(\mathbf{x}_i), c)\right) \sum_{\mathbf{x}_j\in A} d(\mathbf{x}_i, \mathbf{x}_j). \tag{17}$$

Optimiser E^T par rapport à $(A_c)_{c\in C}$ revient en fait à optimiser les E_c^T pour $c \in C$. E_c^T associe à une partie quelconque de Ω une énergie. Pour trouver le minimum de E_c^T sur l'ensemble des parties à q éléments distincts, il suffit de trouver les q éléments de Ω, qui donnent les q plus petites valeurs pour $E_c^T(\{\mathbf{x}\})$. Ceci peut se faire par force brute, c'est-à-dire en calculant $E_c^T(\{\mathbf{x}\})$ pour tout $\mathbf{x} \in \Omega$.

En combinant les étapes présentées au dessus, on obtient l'algorithme 2. Quelques détails doivent être précisés :

- l'initialisation est réalisée par un choix aléatoire des prototypes, c'est-à-dire des sous-ensembles A_c pour $c \in C$, qui sont choisis de sorte à être disjoints deux à deux ;
- lors de l'optimisation de la phase d'affectation, il est possible que deux neurones ou plus réalisent la même énergie minimale, c'est-à-dire qu'on obtienne c et r, tels que $c \neq r$ et $\gamma^T(\mathbf{x}, c) = \gamma^T(\mathbf{x}, r)$. Pour lever l'ambiguïté, on choisi pour $f(\mathbf{x})$ la plus petite valeur pour c qui réalise le minimum de $\gamma^T(\mathbf{x}, c)$;
- de la même façon, il est possible que plusieurs choix pour le prototype A conduisent à la même énergie minimale pour $E_c^T(A)$. On lève l'ambiguïté en conservant les observations d'indice le plus faible et en exigeant que des prototypes associés à des neurones distincts soient disjoints, i.e., $c \neq r$ implique $A_c \cap A_r = \emptyset$.

Notons pour finir que la preuve de convergence proposée dans (Cheng, 1997) s'adapte parfaitement à notre algorithme, même avec T fixé. En effet, si on fixe la structure de voisinage (pour T constant), chaque étape de l'algorithme

Algorithme 2 Les cartes auto-organisatrices pour tableau de dissimilarités

1: Choisir une valeur initiale pour les prototypes $(A_c)_{c\in C}$ {Étape d'initialisation}

2: **Pour** $l = 1$ à L **faire**

3: **Pour tout** élément $\mathbf{x}$ de Ω **faire** {Étape d'affectation}

4: calculer

$$f(\mathbf{x}) = \arg\min_{r\in C} \gamma^T(\mathbf{x}, r)$$

5: **Fin pour**

6: **Pour tout** neurone $c \in C$ **faire** {Étape de représentation}

7: calculer

$$A_c = \arg \min_{A\subset\Omega,\ |A|=q} \sum_{\mathbf{x}_i\in\Omega} K^T\left(\delta(f(\mathbf{x}_i), c)\right) \sum_{\mathbf{x}_j\in A} d(\mathbf{x}_i, \mathbf{x}_j),$$

où $|A|$ désigne le cardinal de l'ensemble A.

8: **Fin pour**

9: **Fin pour**

réduit (au sens large) la valeur de l'énergie E^T. Comme celle-ci est toujours positive, elle converge. De plus, la carte admet un nombre fini de configurations, puisque $\mathcal{R}$ est une liste de M sous-ensembles disjoints de Ω (ensemble de cardinal N) et qu'il existe un nombre fini de partition de Ω en M classes. De ce fait, la valeur limite est atteinte en un nombre fini d'étapes.

En pratique cependant, le voisinage évolue et la convergence n'est pas assurée théoriquement, même si l'expérience montre que l'algorithme se stabilise. De plus, même en cas de structure de voisinage fixée, la configuration finale n'est pas nécessairement celle qui minimise E^T : l'algorithme ne peut que diminuer la valeur initiale. Il est donc préférable de réaliser plusieurs optimisations, en partant de configurations initiales aléatoires distinctes, et de conserver la configuration finale d'énergie minimale.

3.4 Liens avec les travaux antérieurs

Tout d'abord remarquons que si la fonction de voisinage K^T est de la forme $K^T(\mathbf{x}) = 1$ si $\mathbf{x} = 0$ et $K^T(\mathbf{x}) = 0$ sinon, nous retrouvons la méthode de classification de type nuées dynamique d'un tableau de proximité décrite dans (Celeux et al., 1989) (à condition de se limiter à une représentation par un seul prototype, soit $q = 1$). En effet, un tel choix pour K^T revient à ne tenir compte que de la partie de l'énergie définie par l'équation (13). La seconde

partie ($E^T(\mathcal{P}, \mathcal{R})_S$) qui est dépendante de la carte est ignorée. Nous retrouvons aussi, avec cette méthode, les difficultés d'interprétation de l'énergie décrites dans la section 3.2 bien que le critère optimisé soit plus simple.

Deux adaptations des cartes auto-organisatrices aux tableaux de dissimilarités ont été proposés. La première est assez proche de la notre et est due à Kohonen et Somervuo (Kohonen, 1996; Kohonen and Somervuo, 1998, 2002). La seconde est due à Graepel, Burger et Obermayer (Graepel et al., 1998; Graepel and Obermayer, 1999) et est plus éloignée de notre solution.

L'algorithme de Kohonen et Somervuo (Kohonen and Somervuo, 2002) est assez proche de l'algorithme 2. Les différences essentielles sont les suivantes :
- Kohonen et Somervuo associent à chaque neurone un unique prototype ;
- le critère d'affectation utilisé par Kohonen et Somervuo est simplement celui qui associe à une observation le neurone dont le prototype est le plus proche (au sens de la dissimilarité). Ceci pose quelques problèmes. En effet, certaines dissimilarités, comme la distance de Levenshtein (Levenshtein, 1966) entre chaînes de caractères, sont à valeurs entières, ce qui favorise les égalités entre dissimilarités. Il est alors fréquent d'avoir le choix entre plusieurs neurones pour l'affectation d'une observation à un neurone. Réaliser un choix aléatoire entre les différentes possibilités introduit une source d'instabilité dans l'algorithme qui nuit à sa convergence. Somervuo et Kohonen proposent dans (Kohonen and Somervuo, 1998, 2002) de résoudre le problème avec un algorithme d'affectation assez complexe qui consiste à calculer une forme de distance pondérée entre l'observation considérée et les prototypes des neurones voisins du neurone candidat. Le voisinage pris en compte grossit petit à petit tant qu'aucun neurone ne devient un vainqueur unique. En fait, l'algorithme proposé peut être considéré comme une version heuristique de notre critère d'affectation, au moins pour certaines familles de fonctions de voisinage K^T. Notons que notre critère est beaucoup moins sensible au problème de minima multiples ;
- dans la phase de représentation, la recherche du nouveau prototype associé à un neurone c se fait parmi l'ensemble des observations affectées au neurone c (i.e., dans $\mathcal{C}_c$) et aux neurones voisins, alors que nous recherchons le(s) prototype(s) dans l'ensemble des observations. De plus, nous tenons compte des pondérations induites par la structure *a priori* alors que Kohonen et Somervuo minimisent la somme des dissimilarités entre le prototype et les observations affectées au neurone courant et aux neurones voisins, sans pondération. On peut obtenir le même comportement dans notre algorithme en choisissant une famille de fonctions de voisinage K^T un peu particulière.

Le principal défaut de l'algorithme de Kohonen et Somervuo réside dans le fait qu'il ne correspond pas à une optimisation (même heuristique) d'un critère donné. Bien qu'il permette d'obtenir des résultats satisfaisants, il est difficile de savoir exactement ce qu'il fait et donc d'éviter de commettre des erreurs

d'interprétation, par exemple.

L'algorithme de Graepel, Burger et Obermayer est très différent du notre. En revanche, il est aussi basé sur une heuristique d'optimisation d'un critère d'énergie bien défini. Le critère retenu est construit à partir d'une mesure de compacité des classes obtenues, en l'occurrence la somme de toutes les dissimilarités entre les éléments d'une classe (normalisée pour éviter des problèmes liés à des effectifs très différents dans des classes). L'avantage de ce critère est qu'il garantit l'obtention de classes pertinentes, même si la dissimilarité n'est pas métrique. En contrepartie, l'algorithme ne produit pas de prototypes pour résumer une classe, ce qui limite les possibilités en visualisation. De plus, il est très coûteux puisqu'il se comporte en $O(N^2M)$ (pour N observations et M neurones) contre $O(N^2 + NM^2)$ pour le notre (Rossi et al., 2005).

3.5 Exemple de mise en œuvre pour des données simulées

Pour valider le fonctionnement de l'algorithme, nous l'avons testé avec la distance euclidienne (au carré) sur des données simulées. On se donne par exemple un ensemble de 1000 observations disposées uniformément sur un cylindre dans $\mathbb{R}^3$. La structure *a priori* est donnée par une grille bi-dimensionnelle de $21 \times 3 = 63$ neurones.

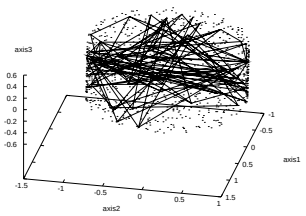

FIG. 1. La carte (21×3 neurones) et le nuage des points

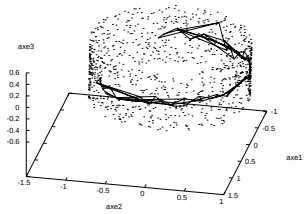

FIG. 2. La carte après 50 itérations

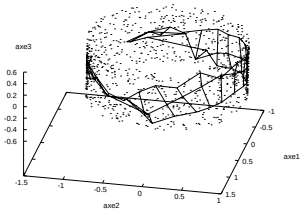

FIG. 3. La carte après 100 itérations

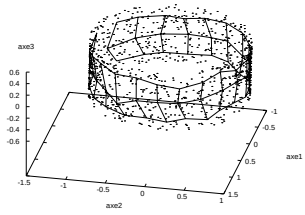

FIG. 4. La carte finale

Dans les figures allant de 1 à 4, nous présentons les données et l'évolution de la carte durant l'apprentissage. La figure 1 présente la carte initiale sur le cylindre après l'initialisation aléatoire des prototypes dans l'ensemble des

observations (on a choisi ici $q = 1$). La carte finale présentée dans la figure 4 montre que la topologie des données à bien été retrouvée par l'algorithme, que la carte est bien déployée et que la quantification est tout à fait satisfaisante.

4 Une application en analyse des usages d'un site Web

4.1 Introduction

Dans cette section, nous proposons une illustration de l'intérêt de notre méthode en l'utilisant pour analyser les usages d'un site Web. Les buts de l'analyse des usages d'un site (le WUM, pour *Web Usage Mining*) sont nombreux (cf Srivastava et al. (2000) pour une présentation synthétique des objectifs principaux du WUM). Nous nous focaliserons ici sur deux aspects, l'analyse des parcours d'un site par ses utilisateurs (les navigations) qui vise à extraire des comportements typiques, et l'analyse de la perception du site qui cherche à faire apparaître les similitudes entre les différents contenus du site au sens de l'utilisation qu'en font les internautes.

Avant de commencer à détailler les étapes de notre application, nous rappelons les principaux concepts propres au WUM (les définitions sont inspirées de celle du W3C et reprises de Tanasa and Trousse 2003)

Le contenu d'un site Web est un ensemble de documents (au sens large) identifiés par des URLs (*Uniform Ressource Locators*, un cas particulier des *Uniform Ressource Identifiers*, Berners-Lee et al. cf 1998). Une URL est de la forme simplifiée suivante : `http://<host>/<path>` (dans cet article, nous ne prendrons pas en compte la partie recherche qui peut terminer une URL). La partie `<host>` correspond au nom DNS du serveur considéré alors que la partie `<path>` correspond au chemin d'accès au document demandé sur le serveur. L'URL `http://www-sop.inria.fr/axis/` correspond ainsi au serveur `www-sop.inria.fr` et au document `axis/` sur ce serveur. Nous ne restreignons pas notre travail à l'analyse d'un site hébergé sur un seul serveur, i.e. d'une partie `<host>` unique. Pour prendre en compte les sites Web complexes utilisant plusieurs serveurs, nous considérons que l'`<host>` peut varier.

La plupart des documents d'un site Web sont des pages au format (X)HTML (W3C HTML Working Group, 2002; Raggett et al., 1999) qui contiennent des hyperliens, c'est-à-dire des références vers d'autres documents accessibles sur le Web (sous forme d'URLs). Une page web regroupe parfois plusieurs documents, par exemple le texte lui-même au format HTML, des images, des scripts externes, une applet Java, etc.

Nous nous intéressons aux utilisateurs d'un site, c'est-à-dire à des personnes qui consultent le site par l'intermédiaire d'un logiciel particulier appelé navigateur. Le navigateur envoie des requêtes HTTP au serveur Web, en désignant les URLs des documents que l'utilisateur souhaite consulter. On appelle **session** l'ensemble des requêtes envoyées à un ou plusieurs serveur(s) web par un utilisateur (par l'intermédiaire de son navigateur). Une session est découpée en **navigations** (ou **visites**). Le découpage est réalisé sur une base temporelle : quand l'écart temporel entre deux requêtes d'une session dépasse un certain seuil (en général 30 minutes), on considère qu'il y a rupture de la navigation. Une navigation est donc une séquence de requêtes séparées d'au plus une certaine durée d'inaction.

4.2 Fichiers log et pré-traitements

Les données d'usage d'un site Web proviennent essentiellement des fichiers log des serveurs concernés. Ceux-ci sont généralement écrits dans le format CLF (*Common Logfile Format*, Luotonen 1995) ou dans sa version étendue qui comporte plus d'informations. Dans ce dernier format, chaque requête vers le serveur Web est représentée par une ligne de la forme suivante (Gaul and Schmidt-Thieme, 2000) :

[Ip] [nom] [login] [date] [requête] [statut] [taille] [referrer] [agent]

Les différents éléments de cette ligne sont les suivants :

Ip	Adresse internet de provenance de la requête (en général, l'ordinateur de l'utilisateur).
nom/login	L'accès à certaines ressources est contrôlé : le fichier log contient alors les identifiants nécessaires pour l'accès.
date	Date et heure de réception de la requête.
requête	La requête reçue par le serveur, dont nous allons essentiellement extraire l'URL du document demandé.
statut	Code numérique précisant le statut de la requête au niveau du serveur (acceptée sans erreur, accès interdit, document inexistant, etc.)
taille	Indique la taille du fichier retourné.
referrer	URL du document dans lequel l'URL demandée a été trouvée (correspond à la structure hyper-textuelle des pages web)

agent Le navigateur et le type de système d'exploitation de l'utilisateur.

Voici un exemple de trace dans un fichier log :

```
194.78.232.8 -- [10/Jan/2003:15:33:43 +0200] "Get /orion/liens.htm
HTTP/1.1" 200 1893 "http://www-sop.inria.fr/orion/index.html"
"Mozilla/4.0 (compatible; MSIE 5.0b1; Mac_PowerPC)
```

On note les informations suivantes :
- 194.78.232.8 est l'adresse internet de l'utilisateur ;
- la requête a été reçue le 10 Janvier 2003 à 15 heures, 33 minutes et 43 secondes ;
- l'URL demandée est `/orion/liens.htm` (il s'agit d'une URL relative à laquelle on doit ajouter le nom du serveur, ici `www-sop.inria.fr`) ;
- la requête a été traitée sans erreur (c'est le statut 200) ;
- le document renvoyé contenait 1893 octets ;
- le lien vers ce document a été trouvé dans le document d'URL `http://www-sop.inria.fr/orion/index.html` ;
- enfin, l'utilisateur travaillait sur un Macintosh (Mac_PowerPC) avec le logiciel Internet Explorer (MSIE) en version 5.0b1.

Malgré leur apparente richesse, les fichiers log sont difficiles à exploiter directement pour diverses raisons, dont voici une sélection :
- un serveur web reçoit en général des requêtes en provenance de plusieurs utilisateurs : les sessions sont donc entremêlées et il faut les reconstruire ;
- les robots d'indexation des moteurs de recherche de type Google parcourent régulièrement la plupart des sites web, ce qui engendre de nombreuses requêtes automatiques : il faut supprimer ces requêtes pour se focaliser sur les requêtes engendrées par des humains ;
- identifier la provenance d'une requête est difficile : certains utilisateurs passent par l'intermédiaire d'un *proxy* qui réalise les requêtes à leur place. Des requêtes provenant d'utilisateurs distincts semblent alors venir d'une même adresse internet (et souvent d'un même agent, celui du *proxy*). De plus, les ordinateurs sont souvent partagés entre utilisateurs dans certains contextes (université, *cyber café*, etc.). Le problème inverse est aussi présent : il est fréquent que les utilisateurs non professionnels disposent seulement d'une adresse internet dynamique qui change donc à chaque connexion. Les requêtes d'un même utilisateur proviennent alors d'adresses internet distinctes ;
- les sites recevant un trafic important utilisent en général plusieurs ordinateurs serveurs, et ont donc plusieurs fichiers log qu'il faut fusionner avant les traitements (Tanasa and Trousse, 2003).

Pour extraire du sens des logs, il est donc indispensable de procéder à une étape de pré-traitement relativement complexe. Nous ne décrirons pas ici les algorithmes retenus : nous avons utilisé les méthodes développées dans notre équipe et décrites dans (Tanasa and Trousse, 2004a,b). Elles permettent d'extraire

des logs multi-serveurs les navigations réalisées par les utilisateurs, après reconstruction de celles-ci et suppression des requêtes engendrées par les robots. Le tableau 1 donne un exemple des données obtenues grâce à ces traitements.

En fait, le pré-traitement peut être vu comme un enrichissement des fichiers log par l'ajout des informations de navigation et de session. Ceci se traduit dans l'exemple de la table 1 par les colonnes portant ces noms qui contiennent des identifiants uniques. L'ensemble des requêtes de la session k est constitué de l'ensemble des lignes dont la colonne session vaut k (de même pour le contenu d'une navigation). Bien entendu, le tableau peut être enrichi par l'ajout d'autres informations disponibles dans les logs comme le *referrer*, l'*agent*, etc. en fonction des besoins de l'analyse.

Les analyses que nous allons décrire dans la suite de l'article sont toutes basées sur un tableau de données de la forme qui vient d'être présentée. Nous avons imposé les contraintes suivantes :
- nous n'analysons pas les requêtes vers des images ;
- nous n'analysons que les requêtes correctes, c'est-à-dire avec un statut compris entre 200 et 399 ;
- le seuil de rupture pour une navigation est de 30 minutes : une navigation est donc une suite de requêtes réalisées par un même utilisateur avec au plus 30 minutes entre deux requêtes ;
- nous ne travaillons pas sur les sessions, mais uniquement sur les navigations : nous ne tenons donc pas compte du fait que plusieurs navigations peuvent provenir d'un même utilisateur. Ceci réduit les problèmes liés aux *proxy*, aux adresses internet dynamiques et au partage d'ordinateurs.

4.3 Prise en compte de la structure du site

Une des difficultés du WUM est qu'un site web de taille moyenne peut contenir des milliers de documents. Même en analysant l'usage du site sur une longue période, il reste difficile d'observer des comportements répétés sur lesquels fonder une analyse : les navigations sont en général très différentes les unes des autres, car chaque utilisateur se focalise sur la partie du site qui l'intéresse. Pour pouvoir analyser le comportement des utilisateurs et leur perception du site étudié, il est donc impératif de simplifier le problème. Tout d'abord, nous supprimons l'aspect temporel des navigations (à l'image de (Mobasher et al., 2002), par exemple) : nous ne tiendrons donc pas compte dans l'analyse du fait qu'une page est visitée avant une autre (l'ordre est souvent une conséquence de la structure du site plutôt qu'un choix délibéré de l'utilisateur).

D'autre part, nous simplifions les navigations en appliquant une solution proposée dans Fu et al. (2000) qui consiste à utiliser la structure hiérarchique

TAB. 1
Tableau de données après pré-traitements

Requête	Navigation	Session	Heure	Date	URL
0	0	0	00:00:05	01/01/03	http://www.inria.fr/
1	0	0	00:00:25	01/01/03	http://www.inria.fr/inria/index.fr.html
2	0	0	00:01:15	01/01/03	http://www.inria.fr/inria/unites.fr.html
3	0	0	00:02:06	01/01/03	http://www.inria.fr/inria/liste-part.fr.html
⋮					⋮
467	73	10	13:21:36	05/01/03	http://www.inria.fr/rapportsactivite/RA94/CROAP.3.4.2.html
468	73	10	13:22:52	05/01/03	http://www.inria.fr/rapportsactivite/RA95/croap/node22.html
469	73	10	13:25:01	05/01/03	http://www.inria.fr/rapportsactivite/RA96/croap/node22.html
⋮					⋮

du site étudié. Une URL est en effet organisée de façon hiérarchique : dans l'URL `http://www-sop.inria.fr/axis/Publications/` choisie sur le site de l'INRIA, on retrouve le serveur de l'unité de recherche de l'INRIA située à Sophia-Antipolis (`www-sop.inria.fr`), le projet de recherche AxIS (`axis`) et la liste de publications de ses membres (`Publications`). Pour simplifier l'analyse d'un ensemble de navigations, on peut donc remplacer les URLs des documents visités par une version "raccourcie" qui se base sur la structure du site.

URL
`http://www-sop.inria.fr/`
`http://www-sop.inria.fr/act_recherche/les_projets_fr.shtml`
`http://www.inria.fr/recherche/equipes/axis`
`http://www-sop.inria.fr/axis/`
`http://www-sop.inria.fr/axis/ra.html`
`http://www.inria.fr/rapportsactivite/RA2003/axis2003/axis_tf.html`

TAB. 2
Une navigation

Considérons par exemple la navigation de la table 2 (nous avons ici extrait la colonne URL d'un tableau de données de la forme du tableau 1). Une simplification possible de cette navigation consiste à ne conserver que le nom du serveur et deux niveaux hiérarchique pour chaque URL, ce qui donne la table 3. En fait, on remplace la variable URL du tableau de données d'origine par plusieurs variables, une pour le serveur, puis une par niveau hiérarchique conservé.

Serveur	Niveau 1	Niveau 2
`www-sop.inria.fr`		
`www-sop.inria.fr`	`act_recherche`	`les_projets_fr.shtml`
`www.inria.fr`	`recherche`	`equipes`
`www-sop.inria.fr`	`axis`	
`www-sop.inria.fr`	`axis`	`ra.html`
`www.inria.fr`	`rapportsactivite`	`RA2003`

TAB. 3
Une représentation simplifiée de la navigation de la table 2

Notons que d'autres méthodes de regroupement d'URLs sont envisageables, en travaillant par exemple sur le contenu des pages ou encore sur la structure d'hyperlien du site. Cependant, il est important de conserver des groupes

facilement interprétables. C'est le cas de notre méthode car le groupe d'URLs est obtenue par un simple élagage de l'arbre associé à l'URL.

4.4 Une analyse de l'usage du site de l'INRIA

4.4.1 Les données

Dans cette section, nous analysons l'usage d'une partie du site Web de l'Institut National de Recherche en Informatique et Automatique (INRIA). Le site de l'INRIA est reparti en plusieurs serveurs dont les rôles sont différents. Le site principal, `www.inria.fr` présente l'institut dans son ensemble, assure la diffusion des rapports de recherche, la promotion de l'institut, etc. Les Unités de Recherche (UR) qui correspondent grossièrement aux différentes implantations géographiques de l'INRIA possèdent aussi des serveurs (il y a six unités de recherche). Nous nous sommes intéressés au serveur de l'UR de Sophia Antipolis, `www-sop.inria.fr`. Comme l'illustre la navigation de la table 2, les différents serveurs de l'INRIA sont étroitement liés et le passage de l'un d'entres eux à un autre se fait de façon totalement transparente pour l'utilisateur. Une analyse multi-serveurs est donc indispensable dans ce contexte.

Nous étudions les accès effectués sur les serveurs pendant les 15 premiers jours de l'année 2003. Nous ne retenons que les longues navigations, c'est à dire les navigations dont la durée est supérieur à 60 secondes et dont le nombre de pages visitées est supérieur à 10 pages. De plus, nous ne nous intéressons qu'aux navigations qui contiennent au moins une requête vers chacun des deux sites. Au total, nous avons donc 3969 navigations qui correspondent à 282552 requêtes valides (statut entre 200 et 399). Dans les analyses, nous appliquons les simplifications décrites dans la section 4.3 en ne conservant que le serveur et le premier niveau de l'URL, que nous désignerons sous le terme de rubrique de niveau 1. Nous obtenons ainsi 196 groupes d'URLs.

Notre analyse basée sur l'adaptation des cartes auto-organisatrices de Kohonen aux tableaux de dissimilarités porte sur deux problèmes distincts à savoir :
– l'analyse et la classification des navigations pour trouver des comportements types d'utilisateurs ;
– l'analyse et la classification des rubriques de niveau 1 pour comprendre et analyser la perception du site par les internautes.

4.4.2 Analyse des navigations

Après la prise en compte de la structure du site, nous obtenons le tableau de données de la table 4, dans laquelle "www" désigne le site principal `www.inria.fr` et "SOP" le site de l'UR de Sophia Antipolis, `www-sop.inria.fr`.

Requête	Navigation	Serveur	Rubrique 1
0	1	www	robotvis
1	1	SOP	robotvis
2	1	www	JGI2002
⋮	⋮	⋮	⋮
282 550	3969	www	freesoft
282 551	3969	SOP	freesoft

TAB. 4
Les logs après simplification

Les observations de notre analyse sont les navigations : il nous faut donc produire un nouveau tableau de données, où chaque ligne contient la description d'une navigation. Comme nous ne tenons pas compte du temps, nous décrivons chaque navigation par deux variables modales. Chaque variable, les rubriques 1 de `www.inria.fr` et de `www-sop.inria.fr`, est maintenant représentée par une distribution de fréquences, comme le montre la table 5.

Navigation	Rubrique 1 sur www	Rubrique 1 sur SOP
1	robovis(10) ; JGI02(0),...	robotvis(15), thesard(36),...
⋮	⋮	⋮
3969	rapport(63), axis(98),...	interne(64), saga(18), ...

TAB. 5
Les navigations sous forme de variables modales

Pour comparer deux navigations, nous utilisons le coefficient d'affinité (Matusita, 1951, 1955; Bacelar-Nicolau, 1985, 2000), dont nous rappelons la définition. Pour chaque variable Y_j à t_j modalités, on suppose données les deux distributions de fréquences, $\delta_{N_i^j} = (n_{ij1}, ..., n_{ijt_j})$ et $\delta_{N_k^j} = (n_{kj1}, ..., n_{kjt_j})$, associées aux navigations N_i et N_k. Le coefficient d'affinité est alors donné par :

$$\text{aff}(\delta_{N_i^j}, \delta_{N_k^j}) = \sum_{l=1}^{t_j} \sqrt{\frac{n_{ijl}}{n_{ij.}} \frac{n_{kjl}}{n_{kj.}}}, \tag{18}$$

avec $n_{ij.} = \sum_{l=1}^{t_j} n_{ijl}$ et n_{ijl} est le nombre d'occurrences pour la navigation i dans la modalité l et la variable Y_j ($1 \leqslant i \leqslant 3969$ et $1 \leqslant l \leqslant t_j$).

Notons que $0 \leqslant \text{aff}(\delta_{N_i^j}, \delta_{N_k^j}) \leqslant 1$. La valeur 1 est atteinte si $\delta_{N_i^j}$ et $\delta_{N_k^j}$ sont identiques ou proportionnelles, alors que la valeur 0 correspond au cas d'orthogonalité.

Considérons maintenant p variables modales (dans notre cas, $p = 2$ et corre-

spond au nombre de serveurs). Soit w_j le poids d'une variable Y_j mesurant son importance, et tel que $0 \leqslant w_j \leqslant 1$ et $\sum_{j=1}^{p} w_j = 1$. Nous définissons la similarité d'affinité pondérée $a(N_i, N_k)$ entre deux navigations N_i et N_k par la moyenne pondérée suivante :

$$a(N_i, N_k) = \sum_{j=1}^{p} w_j \text{aff}(\delta_{N_i^j}, \delta_{N_k^j}) = \sum_{j=1}^{p} w_j \sum_{l=1}^{t_j} \sqrt{\frac{n_{ijl}}{n_{ij.}} \frac{n_{kjl}}{n_{kj.}}} \tag{19}$$

La dissimilarité associée $d(N_i, N_k)$ entre deux navigations N_i et N_k est alors définie comme suit :

$$d(N_i, N_k) = 2(1 - a(N_i, N_k)) \tag{20}$$

Notre algorithme prend en entrée la matrice de dissimilarités basée sur le coefficient d'affinité entre les 3969 navigations. La structure *a priori* est celle d'une grille à deux dimensions, de taille $5 \times 4 = 20$ neurones. Chaque neurone est initialisé par un élément de l'ensemble d'apprentissage Ω choisi aléatoirement (on a donc $q = 1$). Le noyau K est une Gaussienne (cf la section 2.2). La figure 5 représente la carte finale obtenue (le prefixe SOP- signifie que la page a été consultée à partir du site de Sophia).

Chaque case de la carte contient les rubriques de niveau 1 visitées par la navigation du prototype final. Les rubriques indiquées en gras mettent en avant les ressemblances locales. Si nous prenons la classe 1 par exemple, les rubriques 1 en gras sont : "Recherche", "Valorisation", "rrrt" et "rapportactivite". Les voisins directs de la classe 1 sont la classe 2 et la classe 6. Les rubriques "Recherche" et "rapportactivite" ont été visitées par ces trois classes. Les rubriques "rrrt" et "Valorisation" sont communes à la classe 1 et la classe 6. La carte finale obtenue est satisfaisante car les classes voisines partagent quelques rubriques. Il n'y a donc pas lieu d'utiliser pour le paramètre q (le nombre de prototype pour chaque neurone) une valeur strictement supérieure à 1.

Cette première analyse des navigations indique des comportements types des utilisateurs :

- le coin supérieur droit de la carte est consacré aux navigations sur des pages internes de l'INRIA (classes 19 et 20, avec les rubriques 1 "interne","SOP-interne", "semir" un groupe de pages qui décrit les services informatiques internes, etc.) ;
- la classe 18 réalise une transition entre la partie interne et des navigations consacrées à la recherche d'emploi à l'INRIA, c'est-à-dire les classes 11, 12, 13, 16 et 17 (Rubriques 1 "Travailler" et "Recherche") ;
- le coin inférieur gauche de la carte est plutôt consacré à la présentation de la recherche (classes 1, 2, 6 et 7) par l'intermédiaire des rapports d'activités des équipes (rubriques 1 "rapportactivite" et "rrrt"), ainsi que par

Recherche, inria, SOP-act-recherche, Travailler, SOP-axis *Classe 16*	**Recherche, Travailler, SOP-act-recherche, SOP-axis, SOP-semir, actu** *Classe 17*	**Recherche, Travailler, SOP-act-recherche, SOP-semir, SOP-actu, SOP-DR, interne, SOP-interne**, services, DR, Relation-ext *Classe 18*	**interne, SOP-interne, SOP-DR, SOP-semir** *Classe 19*	**interne, SOP-DR, SOP-interne, SOP-semir**, semir, DR *Classe 20*
Recherche, Valorisation, inria, rrrt, actualite, rapportactivite, cgi-bin, act-recherche, Travailler personnel, sinus, SOP-sinus, SOP-miaou, SOP-omega, SOP-smash, SOP-caiman *Classe 11*	**Recherche, Travailler, inria,** SOP-lemme, SOP-Oasis *Classe 12*	**DR, Recherche, Travailler, inria, interne**, personnel, cermics, SOP-cermics, SOP-caiman *Classe 13*	agos-sophia, acacia, SOP-axis *Classe 14*	**publication, dias, SOP-cgi-bin, SOP-dias, SOP-interne**, SOP-actu *Classe 15*
Recherche, Valorisation, rrrt, rapportactivite, SOP-epidaure *Classe 6*	**Recherche, rapportactivite, rrrt, inria**, Robotvis, SOP-Robotvis, SOP-Odyssee *Classe 7*	**rapportactivite, rrrt,** Prisme, SOP-Prisme *Classe 8*	**rrrt, Publications, SOP-cgi-bin, SOP-dias** *Classe 9*	**Publication, SOP-cgi-bin, dias, SOP-dias** *Classe 10*
Recherche, Valorisation, rrrt, rapportactivite, Travailler, presse, personnel, inria, publications, actualite, multimedia, fonctions, SOP-robotvis, SOP-lemme, SOP-mistral *Classe 1*	**Recherche, rapportactivite,** icare, SOP-icare, RA95 *Classe 2*	caiman, SOP-caiman, SOP-glaad, SOP-Safir, SOP-cgi-bin *Classe 3*	chir, SOP-chir, SOP-Saga *Classe 4*	rrrt, icons, SOP-coprin *Classe 5*

FIG. 5. La carte finale 5×4 des navigations. Chaque neurone contient le prototype associé (une navigation référente) et donc la liste des rubriques 1 visitées par cette navigation dans les sites du siège et de Sophia

la communication institutionnelle (rubriques 1 "actualite", "Valorisation", "Recherche", "presse") ;

– le coin inférieur droit de la carte est consacré à des navigations plus ciblées, portant sur des projets de recherche visités depuis leurs rapports techniques et d'activité.

Notons que nous nous sommes focalisés sur les navigations visitant à la fois le site du siège de l'INRIA et celui de l'UR de Sophia. On constate sur la carte que les liens entre les sites semblent fonctionnels. Les navigations de recherche d'emploi (rubrique "Travailler") visitent souvent des sites de projet de recherche de l'UR de Sophia, comme par exemple "SOP-axis" (classe 16), "SOP-sinus", "SOP-Miaou", etc. (classe 11) ou encore "SOP-lemme" et "SOP-Oasis" (classe 12). De façon générale, excepté pour les classes correspondant

à des navigations internes (19 et 20), toutes les navigations référentes passent par des sites de projets de recherche de l'UR de Sophia, ce qui valide le rôle de promotion joué par le site du siège de l'INRIA.

4.4.3 Analyse des rubriques

La seconde analyse concerne les rubriques de niveau 1. Il s'agit de déterminer comment les pages correspondantes sont perçues par les utilisateurs du site. Pour cela, nous construisons un tableau décrivant chaque navigation par la liste des rubriques de niveau 1 consultées. À partir de ce tableau, nous obtenons un tableau binaire dont les individus sont les 196 rubriques de niveau 1 et les variables sont les navigations : pour la rubrique R_j, la variable N_i vaut 1 si et seulement si la navigation N_i a visité au moins une page du groupe d'URLs décrit par la rubrique R_j. Nous obtenons ainsi la table 6.

Navigations / Rubriques	N_1	N_2	...	N_{3969}
R_1 = inria	0	1	...	0
R_2 = Recherche	1	0	...	0
⋮	⋮	⋮	⋮	⋮
R_{196} = SOP-freesoft	0	0	...	0

TAB. 6
Tableau binaire décrivant les 196 rubriques en fonction des navigations

De nombreuses dissimilarités ont été définies pour les tableaux de données binaires. Nous avons retenu celle basée sur l'indice de Jaccard car elle a fait ses preuves dans le cadre de l'analyse de l'usage (cf Foss et al., 2001, par exemple). Nous rappelons la définition de cette dissimilarité.

Considérons deux vecteurs binaires R_1 et R_2 et introduisons les quatre quantités suivantes :
– a est le nombre de j tels que $R_1^j = R_2^j = 1$;
– b est le nombre de j tels que $R_1^j = 0$ et $R_2^j = 1$;
– c est le nombre de j tels que $R_1^j = 1$ et $R_2^j = 0$;
– d est le nombre de j tels que $R_1^j = R_2^j = 0$.
Ces définitions sont résumées par le tableau suivant :

R_2 \ R_1	1	0
1	a	b
0	c	d

L'indice de Jaccard pour les vecteurs R_1 et R_2 est donné par :

$$S(R_1, R_2) = \frac{a}{a+b+c} \tag{21}$$

Dans notre contexte, il correspond à la probabilité de visite de la rubrique R_1 et de la rubrique R_2 sachant qu'on a visité au moins une des deux. La dissimilarité choisie est $(1 - S)$, pour laquelle nous appliquons donc l'algorithme proposé. Nous représentons ainsi 196 rubriques par une structure *a priori* de grille de $4 \times 3 = 12$ neurones.

Afin de faciliter l'analyse des résultats et de montrer leur pertinence, nous avons enrichi la description des groupes d'URLs, c'est-à-dire les rubriques de niveau 1, par une information sémantique obtenue par une analyse humaine du site. Nous avons ainsi pu construire une taxonomie sur les rubriques. Nous avons identifié des rubriques correspondant à des manifestations (colloques, conférences, écoles d'été, etc.) et des rubriques décrivant des projets de recherche. Les autres rubriques ont été classées en rubriques inria ou sophia selon le serveur concerné.

De plus, les projets de recherche de l'INRIA étaient organisés en 2003 selon quatre thèmes :
– thème 1 : réseaux et systèmes ;
– thème 2 : génie logiciel et calcul symbolique ;
– thème 3 : interaction homme-machine, images, données, connaissances ;
– thème 4 : simulation et optimisation de systèmes complexes.
Ces thèmes permettent de subdiviser la catégorie projet.

manifestation	projet (thème 1)	projet (thème 3)	inria
manifestation	projet (thème 1)	projet (thème 4)	projet (thème 2)
projet (thème 2)	projet (thème 4)	projet (thème 4)	projet (thème 4)

FIG. 6. La carte (4× 3) obtenue : chaque case contient l'information sémantique associée à la rubrique référente

Nous présentons d'abord une vue de très haut niveau de la carte obtenue par notre algorithme (voir la figure 6). Celle-ci est obtenue en représentant l'information sémantique associée à la rubrique prototype de chaque neurone. Nous constatons que l'organisation globale de la carte est satisfaisante. En effet, les projets de thème 1 appartiennent à des classes voisines, de même pour les projets de thème 4 ainsi que les manifestations. Il apparaît donc que les utilisateurs du site sont soit convaincus par le regroupement thématique des projets, soit contraints par la structure du site à privilégier des visites communes à des projets dans les mêmes thèmes. En fait, la réponse exacte est un mélange des deux interprétations. Dans les navigations qui s'intéressent à plusieurs projets, il est fréquent de retrouver des pages pivots qui présentent

la recherche à l'INRIA. Ces pages sont organisées par thème et induisent donc naturellement une navigation thématique. Par contre, il est aussi fréquent de trouver des navigations qui touchent plusieurs projets sans passage par des pages pivots. L'aspect thématique est alors induit indirectement par des liens spécifiques (par exemple des publications communes) ou par des ressources externes (par exemple une recherche sur Google qui propose des pages provenant de différents projets).

Nous représentons ensuite sur la figure 7 le contenu partiel de chaque classe. Plus précisément, nous indiquons l'affectation des rubriques de niveau 1 classées dans la catégorie projet. La rubrique prototype est indiquée en gras.

Comme on peut le voir sur la carte détaillée, aucun projet n'a été affecté à la classe 12, classe représentée par la rubrique sémantique "inria". Ceci permet de déduire que la classe 12 est assez homogène. Il apparaît ainsi que les pages des projets sont plutôt visitées indépendamment des autres pages des serveurs.

Nous constatons aussi que les classes 3, 6, 7, 8, 10 et 11 sont composées uniquement de projets appartenant aux même thèmes ce qui permet de déduire que l'adaptation du SOM effectue une bonne quantification de l'espace. Il apparaît aussi que les internautes semblent privilégier des navigations thématiques, comme nous l'avons déjà remarqué pour la vue d'ensemble de la figure 6. Nous pouvons en outre constater pour la classe 11, constituée de projets appartenant au thème 3, la présence simultanée du projet *Aid* et du projet *Axis*. Il faut savoir que le projet *Axis* a remplacé le projet *Aid* au sein de l'INRIA. La visite de l'un entraîne donc très souvent la visite de l'autre, car un lien mutuel existe entre les deux pages. Nous retrouvons le même comportement pour le projet *Odyssee* et le projet *Robotvis*.

Nous constatons la présence de projets dans les classes 5 et 9. En effet, ces deux classes sont représentées par des manifestations et donc la présence des projets permet de déduire que les manifestations sont liées à ces projets. Ce qui explique la visite des pages des projets.

Un dernier point intéressant, si nous revenons à la première carte (figure 6), nous constatons que certains projets de thème 2 appartiennent à des classes très éloignées sur la carte. Pour expliquer ce phénomène, il faut savoir que :
- tout projet à l'INRIA a son propre site Web localisé sur le serveur local de l'unité de recherche à laquelle il est rattaché ;
- de plus, tout projet à l'INRIA possède une page descriptive sur le serveur national du siège.

Prenons l'exemple du projet *cafe*, qui est un projet de l'unité de recherche de Sophia-Antipolis. Son site Web est donc localisé sur le serveur de l'unité de recherche de Sophia, que nous avons noté *SOP-cafe* sur la carte. La page descriptive du projet *cafe* est quant à elle localisée sur le serveur national du

Thème 1 meije Thème 2 Koala, croap Thème 3 odyssee Thème 4 Opale **Classe 9**	Thème 1 SOP-mistral, SOP-Mimosa, SOP-sloop, SOP-rodeo, rodeo, mascotte, **SOP-mascotte**, sloop, SOP-planete, SOP-oasis **Classe 10**	Thème 3 robotvis, epidaure, ariana, acacia, orion, aid, SOP-robotvis, SOP-epidaure, SOP-odyssee, SOP-acacia, **SOP-orion**, SOP-ariana, SOP-aid, SOP-axis, SOP-visa **Classe 11**	**Classe 12**
Thème 1 tropics Thème 3 reves Thème 4 Omega **Classe 5**	Thème 1 Mimosa, tick, **SOP-tick** **Classe 6**	Thème 4 **comore**, mefisto, miaou, SOP-mefisto, SOP-smash **Classe 7**	Thème 2 Prisme, SOP-Prisme, SOP-lemme, **SOP-galaad**, SOP-cafe, SOP-saga, SOP-safir **Classe 8**
Thème 2 cafe, lemme, **certilab** Thème 4 Chir, Fractales, opale **Classe 1**	Thème 1 Mistral, planete, SOP-meije Thème 2 oasis, saga, safir, SOP-Koala Thème 4 **caiman**, sinus **Classe 2**	Thème 4 icare, SOP-sinus, **SOP-icare**, SOP-miaou, SOP-caiman **Classe 3**	Thème 1 SOP-tropics Thème 2 SOP-certilab Thème 3 SOP-reves Thème 4 **SOP-Omega**, SOP-sysdys **Classe 4**

FIG. 7. Affectation des projets aux différents neurones de la carte

siège (elle est désignée par *cafe* sur la carte). Nous nous attendons à ce que ces deux rubriques, *cafe* et *SOP-cafe*, apparaissent dans la même classe ou dans des classes voisines, car elles sont liées sémantiquement. Or, comme on peut le constater sur la carte, la rubrique *cafe* appartient à la classe 1 et la rubrique *SOP-cafe* appartient à la classe 8 qui ne sont pas voisines. Ce phénomène peut être expliqué par l'absence de lien dans la page descriptive vers le site Web du

projet *cafe*. Donc au cours d'une même navigation, l'internaute peut difficilement passer de la page descriptive vers le site Web du projet, ce qui démontre un défaut de conception du site. Nous remarquons le même comportement pour le projet *lemme*.

5 Conclusion

Dans ce travail nous avons proposé une adaptation des cartes auto-organisatrices de Kohonen aux tableaux de dissimilarités. Cette adaptation est basée sur la version *batch* de l'algorithme et permet de traiter tout type de données. Les expériences ont montré l'efficacité de cette méthode et son adaptation aux divers données complexes dès lors que l'on peut définir une mesure de dissimilarité. Cette méthode a aussi donné de bons résultats sur d'autres applications réelles et pour d'autres types de données complexes (voir El Golli, 2004; El Golli et al., 2004).

Remerciements

Nous remercions Brigitte Trousse, Doru Tanasa et Mihai Jurca (équipe AxIS INRIA Sophia-Antipolis) pour leur travail de pré-traitement sur les données analysées dans cet article. Nous remercions aussi le rapporteur anonyme dont les remarques et conseils ont contribué à améliorer le présent article.

Références

Bacelar-Nicolau, H., 1985. The affinity coefficient in cluster analysis. Methods of operations research 53, 507–512.

Bacelar-Nicolau, H., 2000. Analysis of symbolic data : exploratory methods for extracting statistical information from complex data. H. H. Bock and E. Diday, Ch. Similarity and Dissimilarity, pp. 160–165.

Berners-Lee, T., Fielding, R., Masinter, L., August 1998. Uniform Resource Identifiers (URI) : Generic Syntax. RFC 2396, The Internet Society, `http://www.ietf.org/rfc/rfc2396.txt`.

Bock, H. H., 2001. Clustering algorithms and kohonen maps for symbolic data. In : Proc. of The International Conference on New Trends in Computational Statistics with Biomedical Applications (ICNCB).

Bock, H. H., Diday, E., 1999. Analysis of symbolic Data, Exploratory methods for extracting statistical information from complex data. Springer.

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

Chavent, M., De Carvalho, F.A.T.and Lechevallier, Y., Verde, R., 2003. Trois nouvelles méthodes de classification automatique de données symboliques de type intervalle. Revue de Statistique Appliquées 4, 5–29.

Chavent, M., Lechevallier, Y., 2002. Dynamical clustering of interval data. optimization of an adequacy criterion based on hausdorff distance. In : et al., K. J. (Ed.), Classification, Clustering and Data Analysis. Springer, pp. 53–60.

Cheng, Y., November 1997. Convergence and ordering of Kohonen's batch map. Neural Computation 9 (8), 1667–1676.

Cottrell, M., Fort, J.-C., Pagès, G., November 1998. Theoretical aspects of the SOM algorithm. Neurocomputing 21 (1–3), 119–138.

Cottrell, M., Ibbou, S., Letrémy, P., October-November 2004. SOM-based algorithms for qualitative variables. Neural Networks 17 (8–9), 1149–1167.

Cottrell, M., Letrémy, P., January 2005. How to use the kohonen algorithm to simultaneously analyze individuals and modalities in a survey. Neurocomputing 63, 193–207.

De Reyniès, A., Septembre 2002. Classification de données symboliques : une extension de la méthode des nuées dynamiques. In : Actes du IXème congrès de la société Francophone de Classification. pp. 177–180.

De Reyniès, A., 2003. Classification et discrimination en analyse de données symboliques. Thèse de doctorat, Université Paris Dauphine, Paris, France.

Diday, E., 1971. La méthode des nuées dynamiques. Revue statistique appliquée XIX (2), 19–34.

Diday, E., Govaert, G., 1977. Classification automatique avec distances adaptatives. R.A.I.R.O. Informatique Computer Science 11 (4), 329–349.

Diday, E., Simon, J. J., 1976. Clustering analysis. Fu, K. S. (Eds.), Digital Pattern Recognition. Springer, Heidelberg , 47–94.

Dreyfus, G., Martinez, J.-M., Samuelides, M., Gordon, M. B., Badran, F., Thiria, S., Hérault, L., 2002. Réseaux de neurones – méthodologie et applications. Eyrolles, Paris.

El Golli, A., 2004. Extraction de données symboliques et cartes topologiques : Application aux données ayant une structure complexe. Thèse de doctorat, Université Paris-IX Dauphine, Paris, France.

El Golli, A., Conan-Guez, B., Rossi, F., November 2004. Self organizing map and symbolic data. Journal of Symbolic Data Analysis 2 (1).

Fort, J.-C., Cottrell, M., Letrémy, P., 2001. Stochastique on-line algorithm versus batch algorithm for quantization and self-organizing maps. In : Proceedings of Neural Networks for Signal Processing 2001. Falmouth, USA.

Foss, A., Wang, W., Zaïane, O. R., April 2001. A non-parametric approach to web log analysis. In : Proc. of Workshop on Web Mining in First International SIAM Conference on Data Mining (SDM2001). Chicago, IL, pp. 41–50.

Fu, Y., Sandhu, K., Shih, M.-Y., 2000. A generalization-based approach to clustering of web usage sessions. In : Masand, Spiliopoulou (Eds.), Web Usage Analysis and User Profiling. Vol. 1836 of Lecture Notes in Artificial Intelligence. Springer, pp. 21–38.

Gaul, W., Schmidt-Thieme, L., 2000. Frequent generalized subsequences - a problem from web mining. In : Gaul, W., Opitz, O., Schader, M. (Eds.), Data Analysis, Scientific Modelling and Practical Application. Springer, Heidelberg, pp. 429–445.

Graepel, T., Burger, M., Obermayer, K., November 1998. Self-organizing maps : Generalizations and new optimization techniques. Neurocomputing 21, 173–190.

Graepel, T., Obermayer, K., 1999. A stochastic self-organizing map for proximity data. Neural Computation 11 (1), 139–155.

Hammer, B., Micheli, A., Sperduti, A., Strickert, M., March 2004. A general framework for unsupervised processing of structured data. Neurocomputing 57, 3–35.

Heskes, T., Kappen, B., 1993. Error potentials for self-organization. In : Proceedings of 1993 IEEE International Conference on Neural Networks (Joint FUZZ-IEEE'93 and ICNN'93 [IJCNN93]). Vol. III. IEEE/INNS, San Francisco, California, pp. 1219–1223.

Hotelling, H., 1933. Analysis of a complex of statistical variables into principal compenents. Journal of Educational Psychology 24, 417–441, 498–520.

Kohonen, T., 1995, 1997 & 2001. Self-Organizing Maps, 3rd Edition. Vol. 30 of Springer Series in Information Sciences. Springer.

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

Kohonen, T., Somervuo, P. J., 1998. Self-organizing maps of symbol strings. Neurocomputing 21, 19–30.

Kohonen, T., Somervuo, P. J., 2002. How to make large self-organizing maps for nonvectorial data. Neural Networks 15 (8), 945–952.

Levenshtein, V. I., 1966. Binary codes capable of correcting deletions, insertions and reversals. Sov. Phys. Dokl. 6, 707–710.

Luotonen, A., 1995. The common logfile format. `http://www.w3.org/pub/WWW/Daemon/User/Config/Logging.html`.

MacQueen, J., 1965. Some methods for classification and analysis of multivariate observations. In : Proc. of the Fifth Berkeley Symposium on Math., Stat. and Prob. Vol. 1. pp. 281–297.

Matusita, K., 1951. Decision rules based on distance for problems of fit, two samples and estimation. Ann. Math. Stat. 3, 1–30.

Matusita, K., 1955. On the theory of statistical decision functions. Ann. Math. Stat. 26, 631–640.

Mobasher, B., Dai, H., Luo, T., Nakagawa, M., January 2002. Discovery and evaluation of aggregate usage profiles for web personalization. Data Mining and Knowledge Discovery 6 (1), 61–82.

Raggett, D., Le Hors, A., Jacobs, I., December 1999. HTML 4.01 specification. W3C recommandation, W3C, `http://www.w3.org/TR/html4/`.

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

Rossi, F., Conan-Guez, B., El Golli, A., April 2004. Clustering functional data with the som algorithm. In : Proceedings of ESANN 2004. Bruges, Belgium, pp. 305–312.

Rossi, F., Conan-Guez, B., El Golli, A., January 2005. Clustering and visualization of dissimilarity data with a fast self-organizing map. Tech. rep., INRIA, submitted to Pattern Recognition.

Somervuo, P. J., 2004. Online algorithm for the self-organizing map of symbol strings. Neural Networks 17 (1231–1239).

Srivastava, J., Cooley, R., Deshpande, M., Tan, P.-N., 2000. Web usage mining : Discovery and applications of usage patterns from web data. SIGKDD Explorations 1 (2), 12–23.

Tanasa, D., Trousse, B., 2003. Le prétraitement des fichiers log web dans le web usage mining multi-sites. In : Journées Francophones de la toile.

Tanasa, D., Trousse, B., 2004a. Advanced data preprocessing for intersites web usage mining. IEEE Intelligent Systems 19 (2), 59–65.

Tanasa, D., Trousse, B., 2004b. Data preprocessing for wum. IEEE Potentials 23 (3), 22–25.

Tenenbaum, J. B., de Silva, V., Langford, J. C., December 2000. A global geometric framework for nonlinear dimensionality reduction. Science 290 (5500), 2319–2323.

Thiria, S., Lechevallier, Y., Gascuel, O., Canu, S., 1997. Statistique et méthodes neuronales. Dunod, Paris.

Torgerson, W. S., 1952. Multidimensional scaling : I. theory and method. Psychometrika 17, 401–419.

Verde, R., De Carvalho, F., Lechevallier, Y., 2000. A dynamical clustering algorithm for multi-nominal data. In : H.A.L. Kiers, J.-P. Rasson, P. G., Schader, M. (Eds.), Data Analysis, Classification, and Related Method. Springer-Verlag, Heidelberg, pp. 387–394.

W3C HTML Working Group, August 2002. XHTML 1.0 the Extensible HyperText Markup Language. W3C recommandation, W3C, second Edition. `http://www.w3.org/TR/xhtml1/`.

Wang, J.-L., Wang, X., Lin, K.-I., Shasha, D., Shapiro, B. A., Zhang, K., 1999. Evaluating a class of distance-mapping algorithms for data mining and clustering. In : Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 307–311.