Une nouvelle approche visuelle pour la classification hiérarchique et topologique

Hanane Azzag*, Mustapha Lebbah *

* Université Paris 13, LIPN UMR 7030
99, avenue Jean-Baptiste Clément
93430 Villetaneuse, France
prénom.nom@lipn.univ-paris13.fr

Résumé. Nous proposons dans cet article une nouvelle méthode de classification hiérarchique et topologique. Notre approche consiste à construire de manière auto-organisée une partition de données représentées par un ensemble "forêt" d'arbres répartis sur une grille 2D. Chaque cellule de la grille est modélisée par un arbre dont les noeuds représentent les données. La partition globale obtenue est visualisée à l'aide d'une carte de TreeMap dans laquelle chaque TreeMap représente un arbre de données. Nous évaluerons les capacités et les performances de notre approche sur des données aux difficultés variables. Des résultats numériques et visuels seront présentés et discutés.

1 Introduction

Actuellement, la quantité d'informations stockées dans les bases de données est en croissance rapide. A cela se rajoute la complexité de la structure des données à manipuler. En effet ces données ne sont pas uniquement composées d'attributs numériques ou symboliques, mais peuvent être enrichi, par exemple, par des sons, images, vidéos, textes ou des sites Web. Trouver l'information pertinente cachée dans ces ensembles de données structurées est une tâche difficile. Les méthodes de classification et de visualisation sont des techniques qui sont souvent utilisées pour l'analyse des données structurées. L'objectif principal de ce papier est de développer des méthodes qui permettent aux experts du domaine d'analyser de manière intuitive les ensembles de données structurées.

Souvent, l'analyse intuitive des données implique l'utilisation de méthodes de classification hiérarchique combinées à des techniques de visualisation Jain et Dubes (1988); Jain et al. (1999); Handl et al. (2003). Dans la littérature il existe deux types de méthodes de classification hiérarchique : divisives et agglomeratives. Les méthodes divisives sont celles qui partent d'un ensemble d'individus et procèdent par divisions successives des classes jusqu'à l'obtention de la partition la plus fine. En revanche, les méthodes ascendantes partent des singletons et procèdent par agrégations successives. Dans cet article nous proposons une nouvelle approche de classification hiérarchique qui permet de construire des forêts d'arbres et de manière auto-organisée.

Les modèles des cartes auto-organisatrices (Self-Organizing-Map) sont souvent utilisées pour la visualisation et la classification topologique non supervisé Kohonen (2001). Des ex-

tensions et des reformulations du modèle SOM ont été décrites dans Bishop et al. (1998); Fabrice Rossi (2010); Barbara Hammer (2009). Ces approches sont différentes les unes des autres, mais partagent la même idée : représenter de grands ensembles de données par une relation géométrique projetée sur une carte topologique 2D.

En ce qui concerne l'aspect visuel de notre approche, nous pouvons trouver dans la littérature plusieurs algorithmes souvent en 2D, pour la visualisation de structures hiérarchiques. On peut citer les TreeMaps qui transforment de manière récursive un arbre en une série de rectangles, Johnson et Shneiderman (1991); Balzer et Deussen (2005); Shneiderman (1992). Dans un TreeMap, les catégories de données sont représentées par des rectangles imbriqués. L'ensemble de ces éléments est organisé de manière à être représenté totalement dans le plus grand rectangle. Ces rectangles sont conçus afin que leur taille soit proportionnelle au nombre de données formant chaque groupe. Il y a évidemment d'autres manières de représenter une hiérarchie directement sous la forme d'un arbre. Cependant, l'arbre engendré étant de taille généralement conséquente, la visualisation est déformée à l'aide de sortes de loupes, ou d'un affichage hyperbolique 2D, 3D, Carey et al. (2000). La déformation permet à l'ensemble de l'arbre d'être visualisé dans l'espace de représentation en réservant une grande partie de l'affichage à une petite partie de l'arbre et en repoussant le reste des éléments à afficher aux limites de l'espace disponible. On peut également citer les cone-tree de Robertson et al. (1991), qui utilisent comme représentation un arbre conique (Cone-Tree) tridimensionnel dans lequel est associé à chaque noeud parent un sommet du cône, les fils étant ensuite regroupés autour de la base circulaire issue de ce cône. Plus précisément, la racine de l'arbre est le sommet du cône, tous les descendants seront ensuite représentés à l'intérieur de ce cône et à la même hauteur, et ainsi de suite de manière récursive tout en diminuant la base du cône.

Dans cet article, nous introduisons une nouvelle méthode appelée Map-TreeMaps, celle-ci propose une visualisation des données sous forme de carte de TreeMaps et fournit de manière simultanée une classification hiérarchique et topologique. Chaque cellule de la carte représente un arbre de données, le TreeMap associé fournit une vue globale de l'organisation hiérarchique des données sous forme de sous-arbres. La forêt d'arbres obtenue est construite en fonction de règles autonomes basées sur le principe du plus proche voisinage. Le processus topologique proposé est inspiré du modèle SOM et les règles de construction d'arbres sont inspirés d'une méthode biomimétique à base de fourmis artificielles (Azzag et al. (2006b,a)). Le reste de cet article est organisé comme suit : dans la section 2, nous présentons le modèle Map-TreeMaps. Dans la section 3, nous montrons des résultats expérimentaux obtenus sur plusieurs bases de données. Ces bases nous permettent d'illustrer l'utilisation de Map-TreeMaps comme algorithme de classification hiérarchique, topologique et visuelle. Enfin, nous présentons une conclusion et des perspectives sur nos travaux futurs.

2 Modèle de classification hiérarchique et topologique

Le nouveau modèle que nous présentons dans cet article fournit une classification non supervisée et hiérarchique de données dans laquelle chaque partition est une forêt d'arbres organisée sur une carte 2D. Il utilise la même architecture de grille que le modèle classique, associé à la notion de voisinage. Nous avons présenté une première version simplifiée de cette approche dans (Azzag et Lebbah (2010)). Notre modèle consiste à rechercher une classification automatique non supervisée qui fournie une organisation topologique et hiérarchique d'une

base d'apprentissage $A = \{\mathbf{x}_i \in \mathcal{R}^d, i = 1..n\}$ où l'individu $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^j, ..., x_i^d)$. Ce modèle se présente sous forme d'une carte possédant un ordre topologique de n_c cellules. Les cellules sont réparties aux nœuds d'un maillage. Chaque cellule c est la racine d'un sous-arbre $Tree_c$ et chaque noeud $N_{\mathbf{x}_i}$ du sous-arbre représente une donnée $\mathbf{x}_i$. Plus précisément, notre modèle définit un ensemble de sous-arbres répartis sur une grille appelée $\mathcal{C}$.

Prendre en compte la notion de proximité entre sous-arbres dans la carte $\mathcal{C}$, nous oblige à définir une relation de voisinage topologique utilisée dans les cartes topologiques classiques. Ainsi, la topologie de la carte est définie à l'aide d'un graphe non orienté et la distance $\delta(c, r)$ entre deux cellules c et r étant la longueur du chemin le plus court qui sépare les cellules c et r associées aux sous-arbres $Tree_c$ et $Tree_r$. Afin de modéliser l'influence d'une cellule r sur une cellule c (en fonction de leur proximité), on utilise une fonction de voisinage définie à partir d'une fonction noyau $\mathcal{K}$ ($\mathcal{K} \geq 0$ et $\lim_{|y|\to\infty} \mathcal{K}(y_i) = 0$). L'influence mutuelle entre deux sous-arbres $tree_c$ et $tree_r$ de racine c et r sera donc définie par la fonction $\mathcal{K}^T(\delta(c, r))$ où T représente la taille du voisinage (la température). Notons également que chaque sous-arbre est associé à un point représentant noté $\mathbf{w}_c$ qui est une donnée $\mathbf{x}_i$ du sous-arbre $tree_c$ ($\mathbf{w}_c = \mathbf{x}_i \in tree_c$. Choisir un représentant permet d'adapter notre algorithme à n'importe quel type de données. Il suffit juste de définir un tableau de (dis)similarité des données. La qualité de la partition et des sous-arbres associés est définie par la fonction de coût suivante :

$$\mathcal{R}(\chi, \mathcal{W}) = \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{x}_i \in Tree_c} \sum_{r\in C} \mathcal{K}(\delta(\chi(\mathbf{x}_i), r))||\mathbf{x}_i - \mathbf{w}_r||^2 \quad (1)$$

Où χ affecte chaque donnée $\mathbf{x}_i$ à une cellule unique de la carte c.

La minimisation de la fonction de coût $\mathcal{R}$ est un problème d'optimisation combinatoire. Nous proposons par la suite de minimiser la fonction de coût de la même manière que la version "batch", mais en utilisant les caractéristiques statistiques fournies par les sous-arbres (associés à chaque cellule) afin d'accélérer la convergence de l'algorithme. Les trois étapes élémentaires pour la minimisation de la fonction de coût sont définies comme suit :

– **Construction de l'arbre :**
Après chaque affectation d'une donnée $\mathbf{x}_i$ à une cellule c, nous allons chercher à lui trouver la meilleure position dans l'arbre $Tree_c$ associé à cette cellule. Pour cela, nous utilisons des règles d'accrochages/connexions locales inspirées de l'algorithme de classification hiérarchique AntTree, (Azzag et al. (2006a)). La particularité de ces arbres est que chaque noeud N qu'il soit feuille ou noeud interne, représente une donnée $\mathbf{x}$. Nous notons $N_{\mathbf{x}_i}$ le noeud à accrocher associé à la donnée $\mathbf{x}_i$, $N_{\mathbf{x}_{pos}}$ le noeud courant de l'arbre et $N_{\mathbf{x}_{i+}}$ le noeud qui représente la donnée connectée à $N_{\mathbf{x}_{i\,pos}}$ qui est la plus proche en terme de distance à la donnée de $N_{\mathbf{x}_i}$. Nous notons également V_{pos} le voisinage local dans l'arbre perçu par le noeud $N_{\mathbf{x}_i}$ représentant ainsi les autres nœuds (données) connectés à $N_{\mathbf{x}_{pos}}$. Chaque noeud $N_{\mathbf{x}_i}$ représentant ainsi une donnée $\mathbf{x}_i$ sera connecté au noeud $N_{\mathbf{x}_{pos}}$ (position courante), si et seulement si cette action augmente la valeur du seuil de distance $T_{Dist}(N_{\mathbf{x}_{pos}})$. Cette mesure définie la valeur de la distance maximum observée, dans le voisinage local V_{pos}, entre chaque couple de données

connectées au noeud courant $N_{\mathbf{x}_{pos}}$:

$$\begin{aligned} T_{Dist}(N_{\mathbf{x}_{pos}}) &= Max_{j,k}\left\|N_{\mathbf{x}_j} - N_{\mathbf{x}_k}\right\|^2 \\ &= Max_{j,k}\|\mathbf{x}_j - \mathbf{x}_k\|^2 \end{aligned} \quad (2)$$

En d'autres termes, les règles d'accrochages, consiste à comparer un noeud $N_{\mathbf{x}_i}$ au noeud le plus proche $N_{\mathbf{x}_{i+}}$. Dans le cas où les deux nœuds sont suffisamment éloignés entre eux ($\|N_{\mathbf{x}_i} - N_{\mathbf{x}_{i+}}\|^2 > T_{Dist}(N_{\mathbf{x}_{pos}})$), alors le noeud $N_{\mathbf{x}_i}$ sera connecté à la position courante $N_{\mathbf{x}_{pos}}$. Dans le cas contraire, le noeud $N_{\mathbf{x}_i}$ associé à la donnée $\mathbf{x}_i$ sera déplacé vers le noeud le plus proche $N_{\mathbf{x}_{i+}}$. Par conséquent la valeur T_{Dist} diminue pour chaque nœud accroché à l'arbre. En effet chaque connexion d'une donnée $\mathbf{x}_i$ implique la minimisation locale de la valeur du T_{Dist} correspondant.
A la fin de la phase de construction d'arbre, chaque cellule c de la carte $\mathcal{C}$ sera associée à un sous-arbre $tree_c$. Les règles d'accrochage sont basées sur le principe du plus proche voisin. Chaque donnée est donc connectée au plus proche voisin qui formera par la suite son "fils" dans l'arbre.

– **Phase d'affectation par groupe**
Chaque donnée $\mathbf{x}_i$ est connectée dans le sous-arbre $Tree_c$ à un ensemble de données formant ainsi la relation hiérarchique père-fils. Nous utilisons par la suite la fonction *nœudFils*$(\mathbf{x}_i)$ qui fournie l'ensemble des nœuds-fils d'un nœud-parent $N_{\mathbf{x}_i}$ associé à la donnée $\mathbf{x}_i$. A l'itération initiale $t = 0$, *nœudFils*$(\mathbf{x}_i) = \mathbf{x}_i$.

Notre phase d'affectation modifiée consiste à trouver pour chaque donnée $\mathbf{x}_i$ un arbre gagnant, en utilisant la fonction d'affectation χ. Cette cellule sera désignée comme cellule gagnante pour tous les k-plus proches voisins de $\mathbf{x}_i$. En d'autres termes le sous-arbre complet de racine $N_{\mathbf{x}_i}$ est affecté à la cellule gagnante d'une manière récursive. Notons qu'à la première itération les données ne disposent pas de fils. La fonction d'affectation est définie comme suite :

$$\chi(\textit{nœudFils}(\mathbf{x}_i)) = \arg\min_r \sum_{c \in C} \mathcal{K}^T\left(\delta(r,c)\right)\|\mathbf{x}_i - \mathbf{w}_c\|^2 \quad (3)$$

Où $\mathbf{x}_c$ est une donnée $\mathbf{x}_i$ qui représente l'arbre $Tree_c$. Il est possible d'imaginer pour cette phase différentes manières d'affectations.

– **Phase de représentation**
Minimiser $\mathcal{R}$ par rapport à $\mathbf{w}_c$ revient à rechercher le point qui minimise toutes les distances locales.

$$\mathbf{w}_c = min_{\mathbf{w}_c \in tree_c}\left(\sum_{\mathbf{x}_i \neq \mathbf{w}_c ; \mathbf{x}_i \in tree_c} \|\mathbf{x}_i - \mathbf{w}_c\|^2\right), \forall c \in C \quad (4)$$

Interprétation de l'ordre Topologique

Bien que cela n'apparaisse pas explicitement lors des trois étapes précédentes, la température T qui représente la taille du voisinage évolue en fonction des différentes itérations de T_{max} à T_{min}. Dans le cas pratique nous utilisons la fonction de voisinage suivante :

$\mathcal{K}^T(x) = e^{\frac{-\delta(r,c)}{T}}$. Ainsi la décomposition de la fonction de coût $\mathcal{R}$, qui dépend de la valeur de T permet de mettre son expression sous la forme suivante :

$$\begin{aligned}\mathcal{R}^T(\chi,\mathcal{W}) &= \left[\sum_c \sum_{r\neq c} \sum_{\mathbf{x}_i \in tree_r} \mathcal{K}^T(\delta(c,r))\|\mathbf{x}_i - \mathbf{w}_r\|^2\right] \\ &+ \left[\mathcal{K}^T(\delta(c,c)) \sum_c \sum_{\mathbf{x}_i \in tree_c} \|\mathbf{x}_i - \mathbf{w}_c\|^2\right]\end{aligned}$$

où $\delta(c,c) = 0$

Cette fonction de coût est constituée de deux termes. Afin de conserver l'ordre topologique entre les différents arbres, la minimisation du premier terme va permettre de rapprocher les arbres correspondant à deux cellules voisines sur la carte. En effet, si les arbres $tree_c$ et $tree_r$ sont voisins sur la carte, la valeur de $\delta(c,r)$ est alors petite et dans ce cas la valeur de $K^T(\delta(c,r))$ est grande. La minimisation du premier terme aura pour effet de le réduire davantage la valeur de la fonction de coût. Quant à la minimisation du second terme, elle correspond à la minimisation de l'inertie des observations des noeuds accrochés à l'arbre $tree_c$ (dans le cas de l'espace euclidien). Minimiser ce terme revient à appliquer l'algorithme de classification hiérarchique AntTree (Azzag et al. (2006a)).

Nous remarquons également que pour différentes valeurs de la température T, chaque terme de la fonction de coût aura une importance relative dans la minimisation. Pour des grandes valeurs de la température T, le premier terme est prépondérant, dans ce cas, la priorité est de conserver la topologie. Plus la valeur de T est petite, plus le second terme est pris en considération dans la fonction de coût. Dans ce cas, la priorité est de déterminer un représentant des arbres compacts. Il s'agit dans ce cas d'exécuter l'algorithme des AntTree. L'algorithme Map-treemap permet d'obtenir une solution régularisée de l'algorithme de AntTree : la régularisation étant obtenue grâce à la contrainte d'ordre sur les arbres.

3 Validation numérique et visuelle

Nous avons testé et comparé l'algorithme proposé sur plusieurs ensembles de données générées par des lois gaussiennes et uniformes. D'autres bases ont été extraites de Frank et Asuncion (2010) et elles proposent plusieurs difficultés (classification floue, classes déséquilibrées ...). Avant d'étudier les résultats numériques, nous présentons plusieurs visualisations de la carte obtenue avec les TreeMaps associés.

Une des caractéristiques importantes des TreeMaps, est qu'ils utilisent de manière très efficace l'espace d'affichage. Ainsi, il est possible d'afficher de grands arbres avec plusieurs niveaux hiérarchiques dans une quantité minimale d'espace (2D). Les TreeMaps peuvent être particulièrement utiles lorsqu'on dispose de grandes classes. L'affichage d'un arbre à un certain niveau d'abstraction permet de montrer naturellement les informations encapsulées dans l'arborescence et de placer la vue actuelle dans son contexte.

Dans la technique de visualisation que nous proposons, chaque arbre de la cellule est représenté par un TreeMap. Cette initiative vise à obtenir une organisation automatique et globale des arbres sur une carte 2D. La figure 1 montre un exemple d'une forêt comportant quatre structures d'arbres avec les TreeMaps correspondants. Le positionnement des noeuds de chaque arbre dans le TreeMap est un processus récursif. Chaque noeud est représenté par un rectangle de différentes formes. Initialement, les noeuds enfants de la racine sont placés horizontalement sur la zone d'affichage. Ensuite, pour chaque noeud N affiché, ses N_i fils sont placés verticalement sur la zone d'affichage de N. Ce processus est ainsi répété, alternant le positionnement horizontal et vertical jusqu'à ce que tous les noeuds soient affichés. Notons que chaque rectangle a une couleur différente suivant l'étiquette de sa classe réelle . Ceci rend plus facile la comparaison visuelle de la pertinence des groupes obtenus (voir ci-dessous la discussion des résultats visuels).

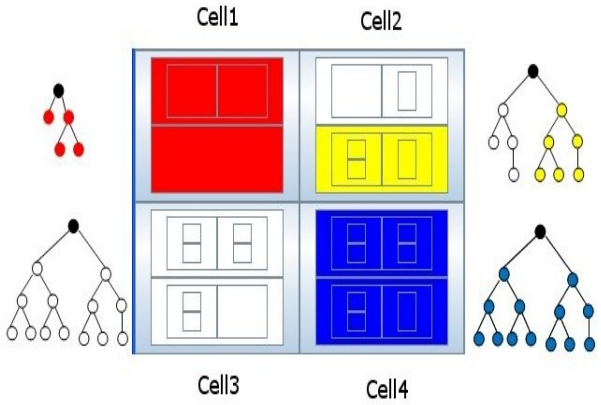


FIG. 1 – *Exemple de forêt d'arbres et les TreeMaps correspondants :* 2×2 *Map-TreeMaps.*

Dans la figure 1 chaque TreeMap représente une organisation hiérarchique des données de l'arbre associé. L'approche Map-TreeMaps que nous proposons a plusieurs propriétés qui permettent une classification simultanée : topologique et hiérarchique. Nous avons élaboré des règles qui respectent les propriétés suivantes : Chaque noeud $N_{\mathbf{x}_i}$ est le plus représentatif de son arborescence. Nous observons dans la figure 1 que les données placées dans le $tree_c$ sont similaires à $N_{\mathbf{x}_i}$ et les noeuds enfants de $N_{\mathbf{x}_i}$ représentent récursivement les sous-arbres qui sont dissimilaire aux sous-arbres frères (sous-arbres du même parent).

3.1 Validation visuelle

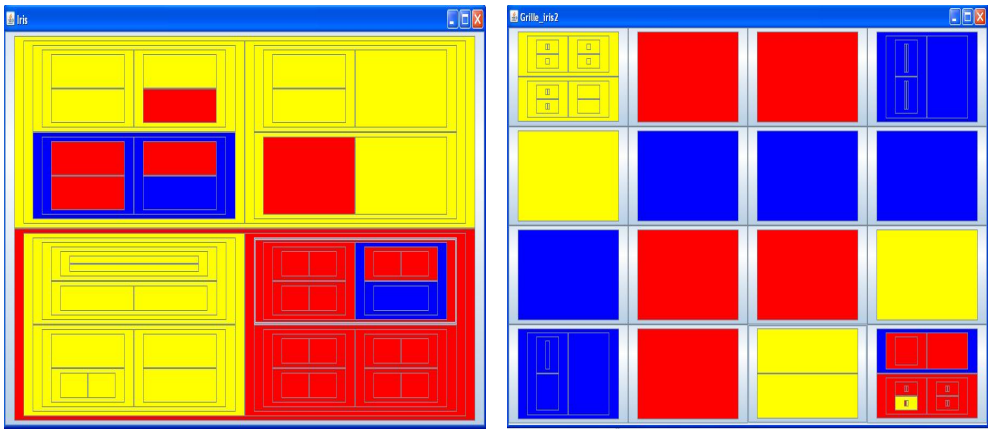
Pour mieux analyser les résultats visuels obtenus, nous avons exécuté l'ensemble des données sur une carte de Map-TreeMaps de 1×1 pour construire un TreeMap unique. Les Figures 2, 3, et 4 représentent des exemples de Map-TreeMaps de dimension 1×1 et 4×4.

En observant les deux cartes générées pour chaque ensemble de données, nous constatons que notre algorithme fournit une carte de TreeMaps qui représente une multi-division du

TreeMap principal. L'organisation topologique et hiérarchique des données est également plus visible sur Map-TreeMaps.

Pour visualiser la cohérence entre l'organisation des données dans l'arbre et l'étiquetage des points, on attribue une couleur à chaque étiquette. Dans chaque figure (2, 3, 4), nous distinguons deux régions sur la carte Map-TreeMaps dédiées aux clusters purs et mélangés. Dans la figure 2.b, nous observons une diagonale allant de la cellule située en haut à droite dans l'image vers la cellule située en bas à gauche dédiée à une seule classe de couleur bleu. Nous pouvons également voir que le TreeMap situé dans la cellule en bas à droite dans l'image est constitué de données de classes différentes. Nous observons dans ce TreeMap, un point jaune placé dans l'arbre, ce résultat est justifié par la présence de données de la classe jaune dans le voisinage. Les mêmes remarques peuvent être présentées pour les bases Lsun et Tetra. Dans la figure 4. Si nous observons le TreeMaps situé dans la cellule en haut à droite dans l'image et celui dans la cellule en bas à gauche, nous pouvons situer le niveau où les clusters se mélangent.

Ce système visuel permet à un expert du domaine de naviguer facilement à travers les données sans avoir à utiliser une méthode de projection. De cette façon, nous espérons être en mesure de parcourir de grandes bases de données, et permettre à l'utilisateur d'interagir facilement avec les données, de percevoir les détails, le contexte et la forme de l'arbre.

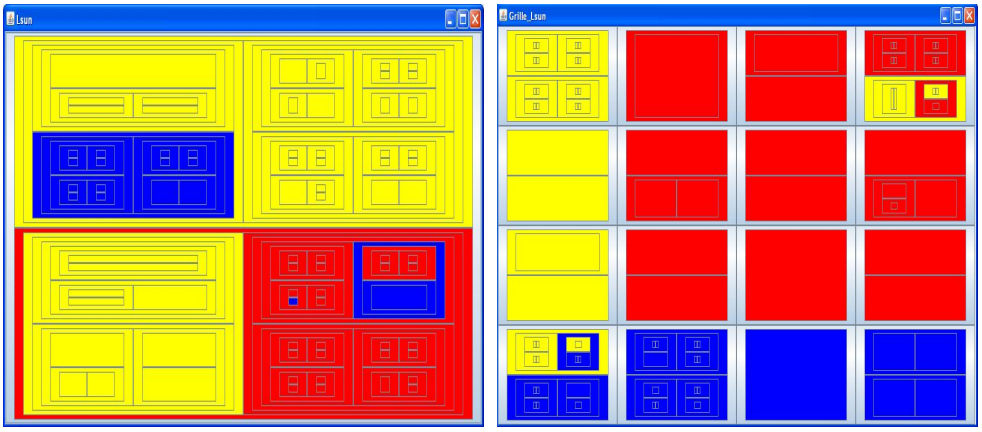


(a) 1×1 TreeMap of data set (b) 4×4 Map-TreeMaps

FIG. 2 – *Iris Dataset*

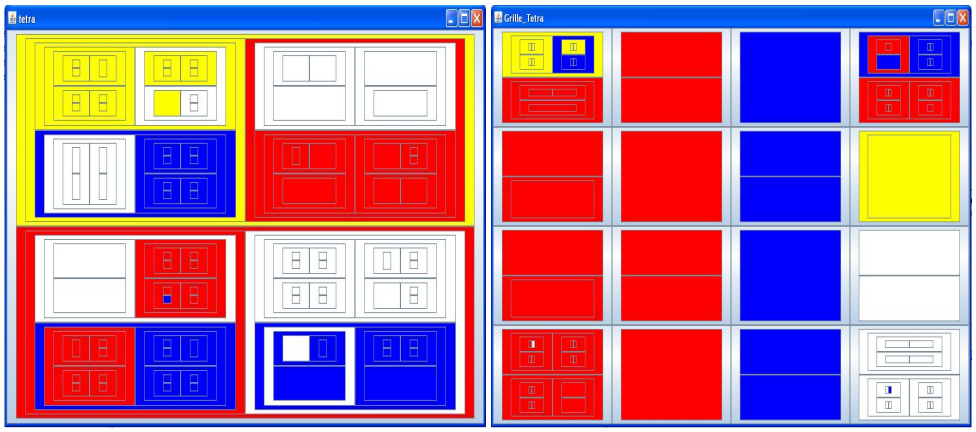
En ce qui concerne les résultats numériques, il nous parait plus objectif de comparer notre modèle avec un algorithme utilisant une architecture similaire. Rappelons ici que le modèle Map-TreeMaps, fournit plus d'informations que les modèles traditionnels type K-moyens ou autres. Dans ce travail nous comparons les résultats obtenus avec ceux du modèle SOM classique. Nous adoptons également les mêmes paramètres : Taille de la carte, voisinage initial, voisinage final

Le tableau 1 présente les résultats numériques obtenus avec Map-TreeMaps et SOM. Nous utilisons deux critères pour mesurer la pertinence du partitionnement obtenu. Le premier est l'indice de Rand qui mesure le pourcentage de paires d'observations appartenant au même



(a) 1×1 TreeMap of data set (b) 4×4 Map-TreeMaps

FIG. 3 – *Lsun Dataset*



(a) 1×1 TreeMap of data set (b) 5×5 Map-TreeMaps

FIG. 4 – *Tetra Dataset*

cluster et ceux n'appartenant pas même cluster, Saporta (2006). Le second indice est le critère Davies et Bouldin (1979) utilisé pour déterminer le nombre optimal des centroïdes K pour K-means. Cet indice proposé par Davies et Bouldin (1979) combine le concept de séparation de cluster et compacité du cluster.

Comme indiqué dans le tableau 1, nous pouvons voir que la méthode Map-TreeMaps fournie des résultats similaires et même meilleurs que ceux obtenus par SOM sur la majorité des cas. En effet, nous pouvons observer que la valeur de l'indice de DB est inférieur à celle de SOM et que l'indice de Rand est plus élevé pour la majorité des cas. Notre objectif à travers cette comparaison est de montrer que Map-TreeMaps obtient les mêmes résultats que SOM et qu'il est comparable aux algorithmes classiques de clustering. Contrairement à la méthode SOM, Map-TreeMaps fournit une vue 'macro' et 'micro' des données.

TAB. 1 – *Les résultats comparatifs obtenus avec Map-TreeMaps et SOM en utilisant les mêmes paramètres (taille de la carte, initialisation de paramètre T)*

bases	taille.	map-treemap		SOM	
		DB	Rand	DB	Rand
Atom(2)	800	1.4	0.88	1.47	0.51
Anneaux (2)	1000	0.80	0.61	0.90	0.51
ART1(4)	400	0.98	0.81	0.85	0.81
Demi-cercle(2)	600	0.58	0.60	0.67	0.5
Glass(7)	214	1.56	0.70	2	0.65
Hepta(7)	212	0.92	0.92	0.85	0.93
Iris(3)	150	1.06	0.75	1.03	0.75
Lsun(3)	400	0.97	0.71	1.09	0.72
Pima(2)	768	1.09	0.5	2.23	0.43
Target(6)	770	1.4	0.85	1.17	0.58
Tetra(4)	400	0.82	0.81	1.25	0.76
TwoDiamonds(2)	800	0.86	0.60	0.81	0.51

La base des chiffres manuscrits

Cette expérience concerne une base de données composées de chiffres manuscrits ('0'−'9') extraits à partir d'une collection de cartes des services HollandaisesFrank et Asuncion (2010). Pour chaque caractère nous avons 200 exemples, nous obtenons ainsi un total de 2000 exemples. Chaque exemple est une imagette binaire (pixel 'noir' ou 'blanc') de dimension 15×16.

La figure 5 montre les chiffres qui sont connectés au premier niveau de chaque arbre associé à la cellule. Nous remarquons qu'un ordre topologique apparaît clairement sur la carte. Cet ordre topologique reste valide pour les autres niveaux dans l'arbre. La figure 6 montre un zoom sur Treemap dans lequel nous pouvons clairement voir le deuxième niveau de certaines cellules de la carte. Nous constatons que l'ordre topologique est toujours maintenu. La même analyse est réalisée sur les autres niveaux des arbres de la foret.

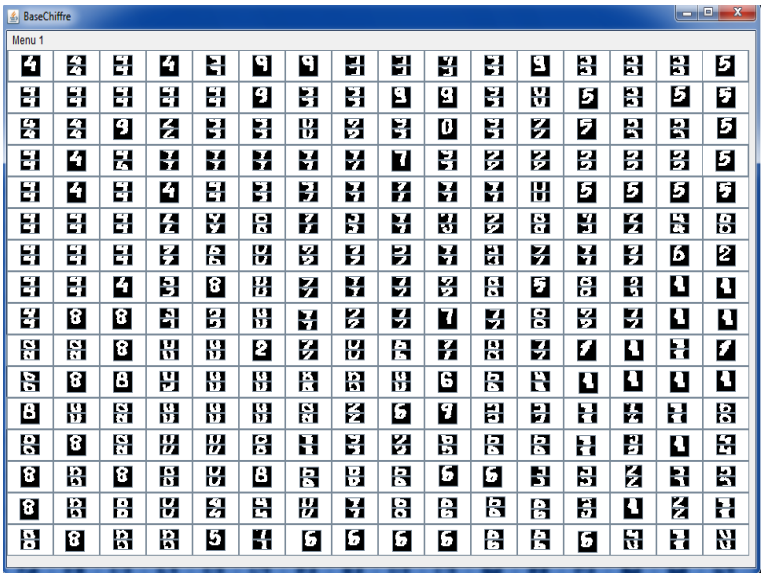


FIG. 5 – *Une vue de la Map-TreeMap au niveau 1.*

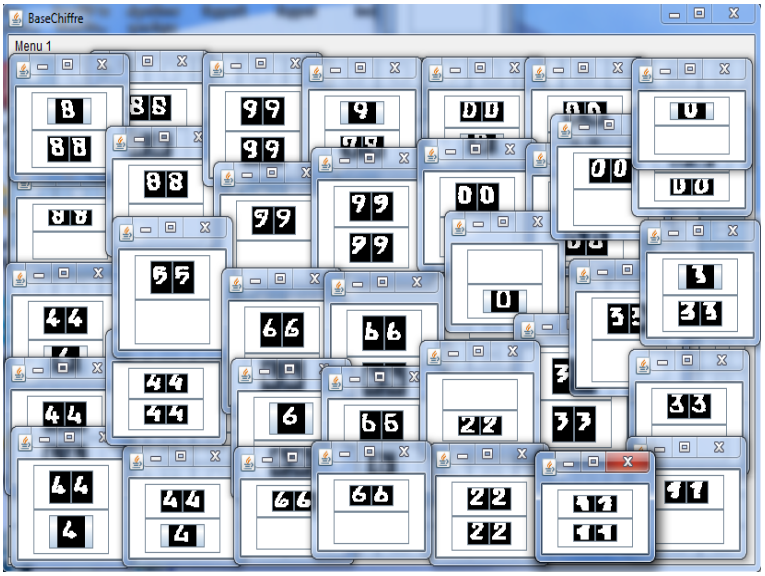


FIG. 6 – *Une vue de la Map-TreeMap au niveau 2.*

4 Conclusions et perspectives

Dans cet article, nous présentons un nouvel algorithme dédié à la classification topologique et hiérarchique qui a les propriétés suivantes : Il fournit une classification hiérarchique des données, qui permet une meilleure visualisation. Il génère à la fois une auto-organisation 2D des arbres associés à chaque cellule et une organisation globale hiérarchique présentée avec les Treemaps. En ce qui concerne les résultats, les expérimentations réalisées prouvent l'efficacité de Map-TreeMaps à traiter des tâches de classification et de visualisation.

Comme perspectives, beaucoup de travail reste à faire. Il est évident que l'utilisation d'arbres permet d'accélérer le processus d'apprentissage Zhang et al. (1996). Ce qui nous permet de traiter/classer des données volumineuses structurées. Dans la phase d'affectation, chaque donnée accroche à son noeud des noeuds fils. Nous voulons généraliser cet algorithme à d'autres types de structures, pas seulement des arbres. Les mêmes principes semblent s'appliquer aussi aux graphes. Enfin, il sera nécessaire de mettre l'accent sur l'aspect visuel de notre approche. En effet, nous comptons améliorer le développement de la vue 2D/3D des différents arbres en treemaps afin de permettre une exploration interactive des données.

Références

Azzag, H., C. Guinot, A. Oliver, et G. Venturini (2006b). A hierarchical ant based clustering algorithm and its use in three real-world applications. In K. S. Wout Dullaert, Marc Sevaux et J. Springael (Eds.), *European Journal of Operational Research (EJOR)*. Special Issue on Applications of Metaheuristics.

Azzag, H., C. Guinot, et G. Venturini (2006a). Data and text mining with hierarchical clustering ants. pp. 153–189.

Azzag, H. et M. Lebbah (2010). Auto-organisation topologique et hiérarchique des données. In S. B. Yahia et J.-M. Petit (Eds.), *EGC*, Volume RNTI-E-19 of *Revue des Nouvelles Technologies de l'Information*, pp. 555–560. Cépaduès-Éditions.

Balzer, M. et O. Deussen (2005). Voronoi treemaps. In *INFOVIS '05 : Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, Washington, DC, USA, pp. 7. IEEE Computer Society.

Barbara Hammer, Alexander Hasenfuß, F. R. (2009). Median topographic maps for biomedical data sets. *CoRR*, 0909.0638.

Bishop, C. M., M. Svensén, et C. K. I.Williams (1998). Gtm : The generative topographic mapping. *Neural Comput 10*(1), 215–234.

Carey, M., F. Kriwaczek, et S. Rüger (2000). A visualization interface for document searching and browsing. In *Proceedings of CIKM 2000 Workshop on New Paradigms in Information Visualization and Manipulation*.

Davies, D. L. et D. W. Bouldin (1979). A cluster separation measure. *IEEE Transactions on Pattern Recognition and Machine Intelligence 1*(2), 224–227.

Fabrice Rossi, N. V.-V. ((2010)). Optimizing an organized modularity measure for topographic graph clustering : A deterministic annealing approach. *Neurocomputing 73(7-9)*.

Frank, A. et A. Asuncion (2010). UCI machine learning repository. Technical report, "University of California, Irvine, School of Information and Computer Sciences, available at : *http ://archive.ics.uci.edu/ml*.

Handl, J., J. Knowles, et M. Dorigo (2003). On the performance of ant-based clustering. pp. 204–213.

Jain, A. et R. Dubes (1988). *Algorithms for Clustering Data*. Prentice Hall Advanced Reference Series.

Jain, A. K., M. N. Murty, et P. J. Flynn (1999). Data clustering : a review. *ACM Computing Surveys 31*(3), 264–323.

Johnson, B. et B. Shneiderman (1991). Tree-maps : A space-filling approach to the visualization of hierarchical information structures. In *Proc. of Visualization'91*, San Diego, CA, pp. 284–291.

Kohonen, T. (2001). *Self-organizing Maps*. Springer Berlin.

Robertson, G. G., J. D. Mackinlay, et S. K. Card (1991). Cone trees : animated 3d visualizations of hierarchical information. In *CHI '91 : Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, pp. 189–194. ACM Press.

Saporta, G. (2006). *Probabilités, analyse des données et statistiques*. Editions Technip.

Shneiderman, B. (1992). Tree visualization with tree-maps : A 2-D space-filling approach. *ACM Transactions on Graphics 11*(1), 92–99.

Zhang, T., R. Ramakrishnan, et M. Livny (1996). BIRCH : an efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pp. 103–114.

Summary

We present in this paper a new clustering method which provides self-organization of hierarchical clustering. This method represents large datasets on a forest of original trees which are projected on a simple 2D geometric relationship using treemap representation. The obtained partition is represented by a map of treemaps, which define a tree of data. In this paper, we provide the rules that build a tree of node/data by using distance between data in order to decide where connect nodes. Visual and empirical results based on both synthetic and real datasets from the UCI repository, are given and discussed.