SoTree : Auto-organisation topologique et hiérarchique des données

Hanane Azzag, Mustapha Lebbah

Université Paris 13, LIPN-UMR 7030 - CNRS
99, avenue Jean-Baptiste Clément
93430 Villetaneuse, France
{hanane.azzag, mustapha.lebbah}@lipn.univ-paris13.fr

Résumé. Nous proposons dans cet article d'introduire une nouvelle approche pour la classification non supervisée hiérarchique. Notre méthode nommée SoTree consiste à construire, d'une manière autonome et simultanée, une partition topologique et hiérarchique des données. Chaque "cluster" de la partition est associé à une cellule d'une grille 2D et est modélisé par un arbre, dont chaque noeud représente une donnée. Nous évaluerons les capacités et les performances de notre approche sur des données aux difficultés variables. Les résultats préliminaires obtenus sont encourageants et prometteurs pour continuer dans cette direction.

1 Introduction

Le problème de la classification de données est identifié comme une des problématiques majeures en extraction des connaissances à partir de données. Depuis des décennies, de nombreux sous-problèmes ont été identifiés, par exemple la sélection de données ou de variables, la variété des espaces de représentation (numérique, symbolique,...), l'incrémentalité, la nécessité de découvrir des concepts, d'obtenir une hiérarchie, etc. La popularité, la complexité et toutes ces variantes du problème de la classification de données (Jain et al. (1999)) ont donné naissance à une multitude de méthodes de résolution. Ces méthodes peuvent à la fois faire appel à des principes heuristiques ou encore mathématiques.

Dans ce travail, les méthodes qui nous intéressent sont celles qui font de la classification topologique et hiérarchique non supervisée de données (Vesanto et Alhoniemi (2000); Vesanto et Sulkava (2002); Ambroise et al. (1998); Golli et al. (2007)). L'avantage des cartes topologiques est de pouvoir représenter et visualiser un grand ensemble de données, ainsi que les regroupements que l'on peut y effectuer. Elles permettent aussi d'utiliser une représentation cartographique visuelle et familière à l'utilisateur.

Dans ce travail nous cherchons à introduire une nouvelle approche de classification simultanée : hiérarchique et topologique nommée SoTree : Self-organizing Tree. L'idée est de déplacer de manière "autonome" des données sur une grille 2D où chaque cellule représente un arbre de données. Nous disposerons ainsi d'une classification horizontale topologique des données sur la grille et d'une classification verticale hiérarchique au niveau de chaque cellule. La fonction topologique de notre algorithme est inspirée des cartes topologiques de Kohonen

(Kohonen (2001); Golli et al. (2007)). La construction de l'arbre est inspirée d'un algorithme de classification hiérarchique biomimétique (Azzag et al. (2006)). Notre article est organisé comme suit : dans la section 2, nous présentons les principes généraux de notre modèle ainsi que les règles autonomes de déplacement sur la carte et de construction de l'arbre. La section 3 est consacrée aux résultats et à l'étude comparative sur des bases de données numériques. La dernière section rassemble les conclusions faites au cours des expérimentations et présente les perspectives.

2 Modèle proposé

Une grande variété d'algorithmes de cartes topologiques est dérivée du premier modèle original proposé par Kohonen. Ces modèles sont différents les uns des autres, mais partagent la même idée de présenter les données de grande dimension en une simple relation géométrique sur une topologie réduite. Le modèle que nous proposons utilise la même architecture de grille, associée à la notion de voisinage. 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 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 :

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

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. En pratique, on se contente d'une solution sous-optimale obtenue à l'aide de la version des nuées dynamiques "batch" de Kohonen (2001). Nous proposons ici 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. (2006). 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 $N_{\mathbf{x}_{ipos}}$ 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*$(\mathbf{x}_i) = \mathbf{x}_i$.

De la même manière que les cartes topologiques classiques, notre phase d'affectation modifiée consiste donc à trouver pour chaque donnée $\mathbf{x}_i$ une cellule dite "gagnante" 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. La fonction d'affectation est définie comme suite :

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

Les propriétés de la distance euclidienne font que les classes obtenues en minimisant la fonction de coût sont compactes et bien séparées : les observations dans un même sous-arbre sont proches les unes des autres (compacité) et sont éloignées des observations des autres sous-arbres (séparation).

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

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

Bien que cela n'apparaisse pas explicitement lors des trois étapes précedentes, la température T évolue en fonction des itérations de T_{max} à T_{min} de la même manière que les cartes topologiques classiques. Dans le cas pratique nous utilisons la fonction de voisinage suivante : $K^T(x) = e^{\frac{-\delta(r,c)}{T}}$.

(1) - Entrées : Carte $\mathcal{C}$ de n_c cellules, base d'apprentissage A , le nombre d'itérations n_{iter}
(2) - Sortie : Carte $\mathcal{C}$ de n_c cellules vides ou constituées de sous-arbres
(3) **pour** $c \in C$ **faire**
(4) $\mathbf{w}_c = \mathbf{x}_i$ /* Initialiser la carte de manière aléatoire */
(5) **finpour**
(6) **pour** $t = 1 : n_{iter}$ **faire**
(7) **pour** $\mathbf{x}_i \in A$ **faire**
(8) - $T = T^{max} \times \left(\frac{T^{min}}{T^{max}}\right)^{\frac{t}{n_{iter}-1}}$
(9) **si** la première affectation de $\mathbf{x}_i$ **alors**
(10) -Rechercher la cellule "gagnante"$\chi(\mathbf{x}_i)$ en utilisant la formule d'affectation (eq. 3)
(11) -Associer la donnée $\mathbf{x}_i$ à un nœud $N_{\mathbf{x}_i}$,
(12) -Accrocher le nœud $N_{\mathbf{x}_i}$ dans le sous-arbre $Tree_{\chi(\mathbf{x}_i)}$, en utilisant les règles de construction d'arbre
(13) -Mettre à jour le point $\mathbf{w}_c$ représentatif en utilisant la formule (eq. 4)
(14)
(15) **sinon**
(16) /* t_{ime} affectation pour la donnée $\mathbf{x}_i$*/
(17) -Rechercher la ième cellule "gagnante"$c_{new} = \chi(\textit{nœudFils}(\mathbf{x}_i))$ en utilisant l'équation 3
(18) **si** $c_{new} \neq c_{old}$ **alors**
(19) -Affecter la donnée $\mathbf{x}_i$ et les fils *nœudFils*$(\mathbf{x}_i)$ à la nouvelle cellule c_{new}
(20) -Accrocher le nœud $N_{\mathbf{x}_i}$ ainsi que les fils associés à l'arbre $tree_{c_{new}}$ en utilisant les règles de construction d'arbre pour chaque donnée.
(21) -Mettre à jour les points $\mathbf{w}_{c_{old}}$ et $\mathbf{w}_{c_{new}}$ représentatifs en utilisant la formule (eq.4)
(22)
(23) **finsi**
(24) **finsi**
(25) **finpour**
(26) **finpour**

ALG 1: Algorithme de classification topographique et hiérarchique : $SoTree$

3 Validation

Afin d'évaluer la qualité de la classification obtenue, nous avons utilisé des bases de données comportant un nombre variable d'observations Blake et Merz (1998) et des données artificielles engendrées par des lois gaussiennes avec des difficultés diverses (recouvrement des classes,

variables non pertinentes, etc.). Avant de comparer nos résultats numériques, nous présentons une visualisation de la carte avec les arbres associés.

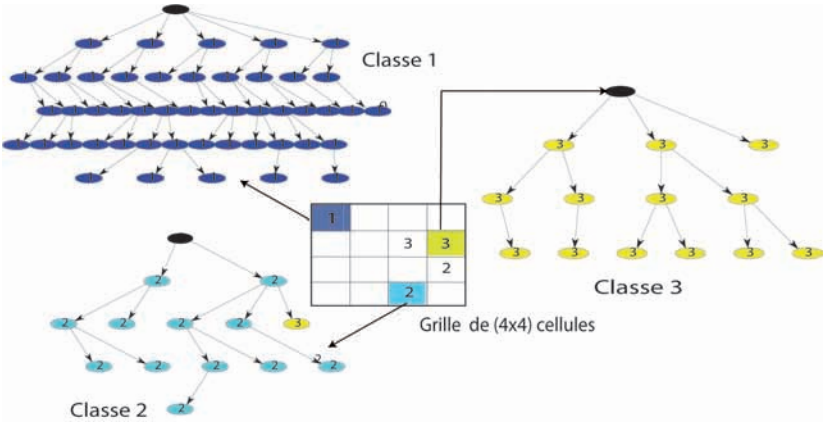

FIG. 1 – *Visualisation des résultats sur la base Iris.*

La figure 1 présente les résultats visuels de notre approche sur la base *Iris*. Les sous-arbres placés directement sur chaque cellule constituent la classification "verticale" fournie par SoTree. Chaque sous-arbre correspond à une classe constituée de toutes les données présentes dans la cellule de la carte. L'approche SoTree répond ainsi aux propriétés visées pour une bonne classification hiérarchique. Avec les règles que nous avons développé, le nœud $N_{\mathbf{x}_i}$ constitue la donnée la plus représentative de son sous-arbre. On observe que les données placées dans $tree_c$ sont les plus proches (similaires) à N_{x_i}. Les données filles (ou nœuds fils) de $N_{\mathbf{x}_i}$ représentent de manière récursive des sous-arbres qui sont les plus éloignés (dissimilaires) possible entre eux.

Bases (Class. Réel.)	Dim.	SoTree	AntTree	CAH
IRIS (3)	150	0.88 (5)	0.94 (5)	0.88 (3)
TWODIAMONDS (2)	800	0.99 (2)	0.99 (7)	0.99 (2)
ART1 (4)	400	0.83 (4)	0.77 (8)	0.84 (5)
ART4 (2)	200	1.0 (2)	0.98 (4)	01.0 (3)
ART6 (4)	400	1.0 (4)	0.93 (4)	01.0 (5)
GLASS (7)	214	0.36 (4)	0.45 (9)	0.49 (3)
THYROID (3)	215	0.73 (3)	0.88 (9)	0.84 (5)

TAB. 1 – *Résultats comparatifs obtenus avec l'approche SoTree, la Classification Ascendante Hiérarchique et l'algorithme AntTree. Le numéro qui suit le taux de bonne classification (taux de pureté) indique le nombre de sous-ensembles de la partition trouvée*

En ce qui concerne les résultats numériques, le tableau 1 résume les résultats préliminaires obtenus. Nous pouvons remarquer que SoTree génère un nombre de classes trouvées très proches du nombre de classes réelles avec des taux de pureté avoisinant le 100%. En comparaison avec les deux autres algorithmes, nous pouvons conclure sur le fait que SoTree est plus performent qu'AntTree et de qualité similaire à la Classification Ascendante Hiérarchique.

4 Conclusions et perspectives

Dans ce travail nous avons développé une nouvelle méthode de classification hiérarchique non supervisée qui possède certaines propriétés : Elle fournit une classification locale hiérarchique des données, ceci permet une meilleure visualisation de l'organisation des données affectées à chaque cellule. Elle fournit une auto-organisation à la fois "verticale" avec les sous-arbres présents dans les cellules et une organisation "horizontale" par la topologie et la prise en compte du voisinage sur la grille. En ce qui concerne les perspectives, les résultats que nous avons présenté sont préliminaires et beaucoup de travail reste à faire. Cependant, lors de la comparaison avec la CAH et AntTree nous avons pu constater que notre nouvelle approche obtient des résultats compétitifs sur plusieurs bases. Il serait également judicieux de s'intéresser à l'aspect visuel de notre approche. En effet nous pensons visualiser en 2D/3D les différents arbres résultants de la classification afin de permettre une exploration interactive des données.

Références

Ambroise, C., G. Séze, F. Badran, et S. Thiria (1998). Hierarchical clustering of self organizing map for cloud classification. *Neurocomputing 30*, 47–52.

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

Blake, C. et C. Merz (1998). Uci repository of machine learning databases. technical report. Technical report, University of California, Department of information and Computer science, Irvine, CA, available at : *ftp ://ftp.ics.uci.edu/pub/machine-learning-databases*.

Golli, A. E., F. Rossi, B. Conan-Guez, et Y. Lechevallier (2007). Une adaptation des cartes auto-organisatrices pour des donnees decrites par un tableau de dissimilarites. *CoRR abs/0709.3586*.

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

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

Refanonyme, R. (2006). Refanonyme refanonyme. *34*.

Vesanto, J. et E. Alhoniemi (2000). Clustering of the self-organizing map. *Neural Networks, IEEE Transactions on 11*(3), 586–600.

Vesanto, J. et M. Sulkava (2002). Distance matrix based clustering of the self-organizing map. In *ICANN '02 : Proceedings of the International Conference on Artificial Neural Networks*, London, UK, pp. 951–956. Springer-Verlag.

Summary

We propose in this paper to introduce a new approach for hierarchical clustering. Our method is called SoTree. SoTree build , independently and simultaneously, a topological and a hierarchical organization of data. Each obtained cluster is associated to cell of a 2D grid and is modeled by a tree, each node represents a given data. We have test the capacity and the performance of our approach on several databases. Preliminary results are encouraging and promising to continue in this direction.