

Editeurs :

Vincent Lemaire - Orange Labs
2 avenue Pierre Marzin, 2300 Lannion
Email : vincent.lemaire@orange.com

Pascal Cuxac - INIST - CNRS
2 allée du Parc de Brabois, CS 10310, 54519 Vandoeuvre les Nancy Cedex
Email : pascal.cuxac@inist.fr

Jean-Charles Lamirel - LORIA - SYNALP Research Team
Campus Scientifique, BP. 239, 54506 Vandoeuvre les Nancy Cedex
Email : jean-charles.lamirel@loria.fr

Publisher:

Vincent Lemaire, Pascal Cuxac, Jean-Charles Lamirel
2 avenue Pierre Marzin
22300 Lannion

Lannion, France, 2015
ISBN : 978-2-9551890-0-9

PRÉFACE

La classification non supervisée ou clustering est de nos jours largement utilisée dans un nombre croissant d'applications dans des domaines aussi divers que l'analyse d'image, la reconnaissance de formes, la fouille de textes, la gestion de la relation client, la veille scientifique ou technologique, la bio-informatique, la recherche d'information, l'analyse de réseaux sociaux...

Bien que le clustering forme un domaine de recherche en soi, avec une longue histoire, et d'innombrables méthodes, de nombreuses questions se posent toujours, telles que par exemple:

- quels sont les bons paramètres : nombre de classes versus finesse d'analyse ?
- comment estimer la qualité d'un clustering, l'importance des variables explicatives ?
- les classes doivent-elles être strictes, floues, ou recouvrantes ?
- comment rendre un clustering robuste et résistant au bruit ?
- comment évaluer l'évolution temporelle du déploiement d'un clustering ?
- ...

L'objectif de cet atelier est de favoriser des présentations et des discussions plutôt que de se focaliser sur des articles écrits complets. La soumission de prises de position bien articulées, d'expériences industrielles et de travaux en cours sont les bienvenus et privilégiés. Des contributions portant sur l'intérêt pratique des travaux, qu'elles viennent de l'industrie ou du monde académique, ou présentant des collaborations entre les deux seraient appréciées.

Le but est le partage d'expérience et de savoir sur les problématiques liées au clustering (coclustering). Le but est aussi de vous (industriels et/ou universitaires) permettre de présenter des problèmes non résolus avec les méthodes de l'état de l'art et/ou les logiciels sur étagères.

V. LEMAIRE
Orange Labs

P. CUXAC
cnrs-inist

J.-CH. LAMIREL
Loria



Membres du comité de lecture

Le Comité de Lecture est constitué de:

Violaine Antoine (ISIMA-LIMOS)
Gilles Bisson (LIG)
Alexis Bondu (EDF RD)
Marc Boullé (Orange Labs)
Laurent Candillier (Expertise lcandillier.free.fr)
Fabrice Clérot (Orange Labs)
Guillaume Cleuziou (LIFO)
Antoine Cornuéjols (AgroParisTech)
Pascal Cuxac (INIST-CNRS)
Patrick Gallinari (LIP6)
Nistor Grozavu (LIPN)
Romain Guigoures (Data Scientist, Zalando)
Pascale Kuntz-Cosperec (Polytech'Nantes)
Nicolas Labroche (Université de Tours)
Jean-Charles Lamirel (LORIA-SYNALP)
Mustapha Lebbah (LIPN)
Vincent Lemaire (Orange Labs)
Jacques-Henri Sublemontier (CEA-LIST)
Fabien Torre (Lille 3)
Christel Vrain (LIFO)

TABLE DES MATIÈRES

Exposé Invité

Panorama des méthodes cherchant un optimum global pour les algorithmes de partitionnement avec contraintes <i>Christel Vrain</i>	1
---	---

Session Exposés

Modèle de Biclustering dans un paradigme "Mapreduce" <i>Tugdual Sarazin, Hanane Azzag, Mustapha Lebbah</i>	5
Clustering conceptuel à base de motifs localement optimaux <i>Frédéric Pennerath</i>	17

Démonstration logiciel

Analyse exploratoire par k-Coclustering avec Khiops CoViz <i>Bruno Guerraz, Marc Boullé, Dominique Gay, Vincent Lemaire et Fabrice Clérot</i> . . .	21
--	----

Index des auteurs	25
--------------------------	-----------

Méthodes déclaratives exactes pour la classification non supervisée sous contraintes

Christel Vrain *

Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022
F-45067, Orléans, France
christel.vrain@univ-orleans.fr

25 janvier 2015

La classification non supervisée sous contraintes (aussi appelée clustering sous contraintes) est une tâche importante de fouille de données, permettant de modéliser plus finement une tâche de clustering en intégrant des contraintes utilisateur. Divers types de contraintes peuvent être considérés ; les contraintes les plus courantes sont les contraintes *must-link* (ML) ou *cannot-link* (CL) spécifiant que des paires d'objets doivent être (respectivement, ne doivent pas être) dans le même cluster. D'autres contraintes peuvent aussi être posées sur les clusters, précisant par exemple, une borne sur leur diamètre ou sur leur taille. Néanmoins, concevoir un algorithme capable de traiter différents types de contraintes est difficile et la plupart des approches tendent à étendre les méthodes classiques de clustering en intégrant un unique type de contraintes.

Plusieurs travaux [11, 10, 3] ont montré l'intérêt de la Programmation par Contraintes (PPC) pour la fouille de données. Modéliser en PPC a deux avantages : la déclarativité qui permet d'ajouter facilement de nouvelles contraintes, la capacité à énumérer toutes les solutions satisfaisant les contraintes et la capacité, dans le cas de l'optimisation d'un critère, à trouver une solution optimale satisfaisant toutes les contraintes (s'il en existe une).

Dans cet exposé, nous ne considérons que l'apprentissage de partitions et nous présentons quelques approches illustrant la modélisation du clustering sous contraintes dans des paradigmes déclaratifs. Nous présentons dans un premier temps une modélisation dans un cadre SAT du clustering [9] permettant d'intégrer des contraintes *must-link* ou *cannot-link* et des contraintes de classe (diamètre des clusters, marge entre clusters). D'autres modélisations SAT existent comme par exemple [13, 2]. Nous donnons deux exemples de l'utilisation de la Programmation par Contraintes pour le clustering sous contraintes. Le premier exemple modélise le clustering conceptuel [12] : les données sont décrites par leurs propriétés et à chaque cluster est associé un motif, i.e. un ensemble de propriétés que toutes les données du cluster satisfont. Dans le second exemple [6, 5, 7, 8], un cadre déclaratif et générique permet de modéliser différentes tâches de clustering sous contraintes, étant donnée une mesure de dissimilarités

*avec la collaboration de Thi-Bich-Hanh Dao et Khanh-Chuong Duong

entre les objets. L'utilisateur peut spécifier un parmi plusieurs critères d'optimisation et combiner différents types de contraintes utilisateur, portant sur les clusters ou sur des paires d'objets. Dans ce modèle, l'utilisateur a la flexibilité de ne pas fixer le nombre de clusters à l'avance, mais seulement de donner une borne supérieure k_{max} et une borne inférieure k_{min} sur le nombre de clusters. Le modèle trouve, s'il existe une solution, une partition en k classes, avec $k_{min} \leq k \leq k_{max}$, satisfaisant toutes les contraintes et optimisant globalement le critère spécifié. Il est bien connu que l'efficacité d'un modèle fondé sur la PPC dépend fortement du choix des variables qui représentent les éléments du problème et du choix des contraintes utilisées. Pour exploiter pleinement la puissance de la PPC, il est également nécessaire de renforcer la propagation de contraintes et la capacité à casser les symétries (différentes représentations d'une même solution). Nous montrons que des améliorations du modèle reposant sur des résultats théoriques permettent de l'alléger sans changer la sémantique. De même, le développement d'algorithmes de filtrage adaptés permet d'améliorer l'efficacité.

Le critère d'optimisation a un fort impact sur l'efficacité des modèles. Les critères considérés sont souvent la minimisation du diamètre maximum, la maximisation de la marge ou la somme des dissimilarités intra-clusters (WCSD). Pour terminer, nous évoquerons des méthodes exactes [4, 1] s'attaquant au critère WCSS, minimisant la somme des carrés des distances au centre des clusters, critère utilisé dans l'algorithme des k-moyennes .

Références

- [1] Behrouz Babaki, Tias Guns, and Siegfried Nijssen. Constrained clustering using column generation. In *Proceedings of the 11th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 438–454, 2014.
- [2] J. Berg and M. Jarvisalo. Optimal Correlation Clustering via MaxSAT. In *Proceedings of the 13th IEEE International Conference on Data Mining Workshops*, pages 750–757, 2013.
- [3] Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, Samir Loudni, and Jean-Philippe Métivier. Discovering Knowledge using a Constraint-based Language. *CoRR*, abs/1107.3407, 2011.
- [4] M.J. Brusco. A repetitive branch-and-bound procedure for minimum within-cluster sum of squares partitioning. *Psychometrika*, pages 347–363, 2006.
- [5] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. A Declarative Framework for Constrained Clustering. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 419–434, 2013.
- [6] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Un modèle général pour la classification non supervisée sous contraintes utilisateur. In *Neuvième Journées Francophones de Programmation par Contraintes*, 2013.
- [7] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Classification non supervisée mono et bi-objectif sous contraintes utilisateur par la programma-

- tion par contraintes. In *Dixième Journées Francophones de Programmation par Contraintes*, 2014.
- [8] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Clustering sous contraintes en ppc. *Revue d'Intelligence Artificielle*, 28(5) :523–545, 2014.
 - [9] Ian Davidson, S. S. Ravi, and Leonid Shamis. A SAT-based Framework for Efficient Constrained Clustering. In *Proceedings of the 10th SIAM International Conference on Data Mining*, pages 94–105, 2010.
 - [10] L. De Raedt, T. Guns, and S. Nijssen. Constraint Programming for Data Mining and Machine Learning. In *Proc. of the 24th AAAI Conference on Artificial Intelligence*, 2010.
 - [11] Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint programming for itemset mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 204–212, 2008.
 - [12] Tias Guns, Siegfried Nijssen, and Luc De Raedt. k-Pattern set mining under constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(2) :402–418, 2013.
 - [13] Jean-Philippe Métivier, Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, and Samir Loudni. Constrained Clustering Using SAT. In *Proceedings of the 11th International Symposium on Advances in Intelligent Data Analysis*, pages 207–218, 2012.

Modèle de Biclustering dans un paradigme "Mapreduce"

Tugdual Sarazin* Mustapha Lebbah*, Hanane Azzag *

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

Résumé. Le Biclustering est une tâche principale dans plusieurs domaines de l'apprentissage automatique. Actuellement les données utilisées sont de plus en plus complexes et les méthodes de clustering traditionnelles bien qu'efficaces nécessitent de mettre en place de nouvelles plateformes pour gérer ces données. Dans ce travail nous proposons une nouvelle approche de biclustering nommée BiTM basée sur les cartes auto-organisatrices SOM. Nous proposons une implémentation sous le paradigme MapReduce permettant le passage à l'échelle. Les expérimentations réalisées sous la plateforme Spark ont montrées les bonnes performances de notre modèle avec des accélérations linéaires.

1 Introduction

Le BiClustering consiste à réaliser un clustering simultanément sur les observations et les variables. Traditionnellement les termes co-clustering, bidimensional clustering et subspace clustering sont souvent utilisés pour désigner la même problématique. Le Biclustering fournit des modèles locaux représentant à la fois des sous-ensembles d'observations et de caractéristiques similaires. Il existe dans la littérature plusieurs approches de biclustering tel que la méthode NBVD, proposé dans (Long et al., 2005), qui consiste à factoriser la matrice de données en trois composants : les coefficients appliqués aux lignes de la matrice, les blocs de valeurs de la matrice et les coefficients appliqués aux colonnes.

Govaert et al ont introduit une adaptation de l'algorithme k -means au biclustering nommée "Croec" qui permet de découvrir tous les biclusters en même temps. Elle procède par optimisation des partitions de lignes et de colonnes en réalisant la procédure itérative du k -means. Cette adaptation (Govaert, 1983) a été déclinée en trois algorithmes pour les données continues, binaires et les tableaux de contingences.

Dans (Labioud et Nadif, 2011), les auteurs ont proposés une approche nommée "Co-clustering Under Nonnegative Matrix Tri-Factorization" (CUNMTF), qui généralise le concept de la NMF (Lee et Seung, 1999) en factorisant la matrice d'origine en trois matrices non-négatives. Les auteurs montrent que le double k -means est équivalent au problème algébrique de la NMF sous certaines contraintes. D'autres modèles probabiliste de biclustering sont proposés dans (Govaert et Nadif, 2008; Priam et al., 2008).

Le Biclustering a de nombreuses applications et devient un challenge de plus en plus important avec l'augmentation des volumes de données. Cependant les bons algorithmes de clus-

biclustering

tering sont encore extrêmement utiles, il est donc nécessaire de les adapter aux nouvelles architectures massivement distribuées utilisant le paradigme MapReduce (Lv et al., 2010; Lin et al., 2011).

Le paradigme de programmation MapReduce (Dean et Ghemawat, 2008) et l'écosystème qui en découle sont actuellement l'une des solutions les plus adaptées au traitement des larges volumes de données. Durant les dernières années le paradigme de programmation MapReduce s'est imposé comme l'une des solutions les plus utilisées pour le traitement de grands volumes de données. De nombreux développeurs ont ainsi migrer vers ce nouveau paradigme (Ene et al., 2011; Sul et Tovchigrechko, 2011; Ferreira Cordeiro et al., 2011; Ghoting et al., 2011). Le modèle de programmation MapReduce est simple, il permet au développeur de s'affranchir des problématiques de gestion de la mémoire et de communication entre les processus / machines. De plus un algorithme MapReduce est agnostique au volume des données traité (Karloff et al., 2010). Le développement d'un programme MapReduce consiste à écrire une ou plusieurs fonctions primitives Map et Reduce. Les fonctions de Map servent généralement à extraire l'information utile d'une partie des données (phrase d'un texte, vecteur d'une matrice, ...). Elles fournissent en sortie un ou plusieurs couples de clé-valeur. Les fonctions de Reduce sont généralement utilisées pour agréger les données en sortie de la fonction de Map. Elles ont en entrée une clé et l'ensemble des valeurs associées à cette clé. Les fonctions de Map et Reduce peuvent être exécutées de façon autonomes sur toutes les machines du cluster simultanément. L'architecture MapReduce se charge de leur répartition et du transfert des données à l'entrée et à la sortie des fonctions. Hadoop¹ est sûrement l'une des architectures MapReduce les plus populaire pour le traitement des gros volumes de données. Bien qu'Hadoop est démocratisé et simplifié les problématiques de traitement et d'analyse des gros volumes ça reste une méthode peu performante dans le domaine de l'apprentissage automatique. En effet la majorité des algorithmes d'apprentissage - et plus particulièrement les algorithmes de Clustering - lisent plusieurs fois les données afin d'optimiser un paramètre or cette lecture des données est assez lente sur une architecture Hadoop. Le framework de traitement de données distribué Spark² (Zaharia et al., 2010) résout ce problème en permettant de stocker les données en mémoire. Lors de l'exécution d'un algorithme d'apprentissage les données sont chargées en mémoire (depuis le disque) dès le début de l'apprentissage, par la suite elles seront relues depuis la mémoire. Ainsi, les gains de temps sont très significatifs lors des nombreuses itérations.

Dans ce papier nous proposons une nouvelle approche globale de biclustering basé sur les cartes auto-organisatrices et le calcul distribué. Le modèle de biclustering BiTM (Biclustering using Topological Maps) a déjà donnée lieu à une publication dans (Chaibi et al., 2014). Dans ce papier nous proposons une adaptation de ces travaux a l'architecture MapReduce avec une implémentation de BiTM sous Spark, une technologie open source de calcul distribué incluant plusieurs paradigmes de programmation. Les principales problématiques abordées dans ce papier sont la minimisation de la fonction et la taille des données en entrée et en sortie des fonctions primitive (Map et Reduce) d'un algorithme de biclustering topologique.

La suite du papier est organisée comme suit : la section 2 décrit plusieurs travaux antérieurs liés à cette problématique. Dans la section 3 nous rappelons l'approche de biclustering basée sur les cartes topologique (ou cartes auto-organisatrices). Dans la section 3.2, nous décrivons

1. www.hadoop.com
2. <http://spark-project.org/>

la décomposition en fonction élémentaire Map et Reduce. La section 4 présente les résultats expérimentaux. Pour finir, la section 5 conclut et présente nos futures travaux de recherche.

2 Travaux précédents

Les paradigmes de programmation existants pour la parallélisation à grande échelle tel que MapReduce et MPI (Message Passing Interface) ont été fréquemment utilisés pour implémenter des algorithmes d'analyse de données. MapReduce est le paradigme le plus utilisé dans le traitement de larges volumes de données. De plus il est souvent utilisé dans un écosystème Hadoop qui gère automatiquement l'accès, la réplication et la distribution des données. Cependant, ces paradigmes de programmation parallèles sont de trop bas niveau et mal adaptés pour la mise en oeuvre d'algorithmes d'apprentissage automatique. Pour répondre à cette problématique l'auteur de (Ghoting et al., 2011) propose une infrastructure portable spécialement conçu pour la mise en oeuvre rapide d'algorithmes d'apprentissage automatique distribués. Récemment une bibliothèque MapReduce-MPI a été mise à disposition par Sandia Lab pour faciliter le portage d'applications séquentielles vers des infrastructures de Calcul Haute Performance (HPC). En se basant sur cette bibliothèque l'auteur de (Sul et Tovchigrechko, 2011) a créé deux applications de bioinformatique open source. Dans Ferreira Cordeiro et al. (2011) l'auteur utilise aussi le paradigme MapReduce. Il constate que les principales problématiques de cette architecture sont : comment minimiser les E/S (entrée/sortie) en prenant en compte la répartition des données dans le système et comment minimiser les communications réseaux entre les noeuds.

Afin de répondre aux problématiques du BigData, de nouveaux paradigmes de programmation distribuée sont apparus tel que MapReduce, Pregel, and PowerGraph Malewicz et al. (2010); Low et al. (2012). Les solutions de biclustering distribués sont essentielles lorsque les données sont réparties sur un grand nombre de machines, comme c'est le cas dans les architectures BigData actuelles. La majorité des travaux de biclustering dans un contexte BigData proviennent surtout du domaine de l'analyse génomique Liu et Chen (2008) et du filtrage collaboratif George et Merugu (2005). Cependant une approche de Biclustering nommée Disco basée sur le paradigme MapReduce a été proposée dans Papadimitriou et Sun (2008). Dans cet article les auteurs introduisent une approche pratique de pré-traitement des données et de co-clustering. Notre approche est très différente, car elle est basée sur une représentation topologique des clusters grâce aux cartes auto-organisatrices Kohonen (2001). Une autre différence entre nos deux approches est le framework utilisé, DisCo est basé sur Hadoop MapReduce tandis que BiTM est implémenté avec Spark.

3 Le Modèle BiTM

3.1 BiTM Model

Dans ce document, les matrices sont notées par des lettres majuscules en gras tel que **H**. Les vecteurs sont notés par des lettres minuscules en gras tel que **g** et les éléments des matrices

biclustering

et des vecteurs sont représentés par des lettres minuscule tel que g_i^j et h_i^j .

De même que les cartes auto-organisatrice traditionnelles, BiTM (Biclustering using Topological Maps) est constitué d'un ensemble discret de cellules \mathcal{C} appelées carte avec K cellules.

La carte a une topologie discrète définie comme un graphe non orienté, c'est habituellement une grille en deux dimensions. Pour chaque couple de cellules (c, r) de la carte, la distance $\delta(c, r)$ est définie comme la longueur de la chaîne la plus courte reliant les cellules r et c sur la grille. Soit \mathcal{R}^d l'espace de données euclidien et \mathbf{D} la matrice de données, où chaque observation $\mathbf{x}_i = (x_i^1, x_i^2, \dots, x_i^j, \dots, x_i^d)$ est un vecteur dans $\mathbf{D} \subset \mathcal{R}^d$. L'ensemble des lignes (observations) est désignée par $I = \{1, \dots, N\}$. De même, l'ensemble des colonnes (variables) est désignée par $J = \{1, \dots, d\}$. Nous nous intéressons au clustering simultané des observations I en K clusters $\{P_1, P_2, \dots, P_k, \dots, P_K\}$, où $P_k = \{\mathbf{x}_i, \phi_z(\mathbf{x}_i) = k\}$ et des variables J en L clusters $\{Q_1, Q_2, \dots, Q_l, \dots, Q_L\}$ ou $Q_l = \{\mathbf{x}^j, \phi_w(\mathbf{x}^j) = l\}$. La fonction d'affectation des lignes (observations) est noté ϕ_z et La fonction d'affectation des colonnes (variables) est noté ϕ_w .

L'objectif principal de BiTM est de transformer une matrice de données \mathbf{D} en une structure de blocs organisés sur une carte topologique. Dans BiTM chaque cellule $r \in \mathcal{C}$ est associée à un prototype $\mathbf{g}_k = (g_k^1, g_k^2, \dots, g_k^l, \dots, g_k^L)$, ou $L < d$ et $g_k^l \in \mathcal{R}$.

Afin de faciliter la compréhension, nous définissons deux matrices binaires $\mathbf{Z} = [z_i^k]$ et $\mathbf{W} = [w_j^l]$ sauvegardant respectivement l'affectation des observations et des variables :

$$z_i^k = \begin{cases} 1 & \text{if } \mathbf{x}_i \in P_k, \\ 0 & \text{else} \end{cases}$$

$$w_j^l = \begin{cases} 1 & \text{if } \mathbf{x}^j \in Q_l \\ 0 & \text{else} \end{cases}$$

Pour regrouper \mathbf{D} en K clusters d'observations et en L clusters de variables, nous proposons la nouvelle fonction objectif afin d'optimiser le processus de biclustering :

$$\mathcal{R}(\mathbf{W}, \mathbf{Z}, \mathbf{G}) = \sum_{k=1}^K \sum_{l=1}^L \sum_{i=1}^N \sum_{j=1}^d \sum_{r=1}^K \mathcal{H}^T(\delta(r, k)) z_i^k w_j^l (x_i^j - g_r^l)^2 \quad (1)$$

Nous pouvons détecter le bloc ou le bicluster de données désignés par $B_k^l = \{(x_i^j | z_i^k \times w_j^l = 1)\}$. $\mathbf{G} = \{\mathbf{g}_1, \dots, \mathbf{g}_k\}$ désigne l'ensemble des prototypes. La fonction de voisinage $\mathcal{H}^T(\delta) = \mathcal{H}(\delta/T)$ est une fonction positive qui diminue au fur et à mesure que la distance entre l'espace latent entre deux cellules \mathcal{C} augmente et lorsque la largeur de la fonction de voisinage T augmente. Ainsi T diminue entre l'intervalle T_{max} à T_{min} . Dans la pratique, nous utilisons la fonction de voisinage $\mathcal{H}^T(\delta(c, r)) = \exp\left(\frac{-\delta(c, r)}{T}\right)$ et $T = T_{max} \left(\frac{T_{min}}{T_{max}}\right)^{\frac{t}{t_f - 1}}$, où t est la période actuelle et t_f le nombre de période.

La fonction objectif (Eq. 1) peut être localement minimiser en résolvant itérativement les trois problèmes de minimisation suivants :

- Problème 1 : Fixer $\mathbf{G} = \hat{\mathbf{G}}$ et $\mathbf{W} = \hat{\mathbf{W}}$, résoudre le problème réduit $\mathcal{R}(\hat{\mathbf{W}}, \mathbf{Z}, \hat{\mathbf{G}})$;
- Problème 2 : Fixer $\mathbf{G} = \hat{\mathbf{G}}$ et $\mathbf{Z} = \hat{\mathbf{Z}}$, résoudre le problème réduit $\mathcal{R}(\mathbf{W}, \hat{\mathbf{Z}}, \hat{\mathbf{G}})$;

— Problème 3 : Fixer \mathbf{W} et \mathbf{Z} , résoudre le problème réduit $\mathcal{R}(\hat{\mathbf{W}}, \hat{\mathbf{Z}}, \mathbf{G})$;

Afin de réduire le temps de calcul, nous affectons chaque observation et variable sans utiliser la fonction de voisinage contrairement au carte topologique classique.

Le problème 1 est résolu en définissant z_k^j tel que :

$$z_i^k = \begin{cases} 1 & \text{if } \mathbf{x}_i \in P_k, k = \phi_z(\mathbf{x}_i) \\ 0 & \text{else} \end{cases}$$

où chaque observation \mathbf{x}_i est affectée au plus proche prototype \mathbf{g}_k en utilisant la fonction d'affectation définie comme suite :

$$\phi_z(\mathbf{x}_i) = \arg \min_c \sum_{j=1}^d \sum_{l=1}^L w_j^l (x_i^j - g_c^l)^2 \quad (2)$$

Le problème 2 est résolu en définissant w_k^j tel que :

$$w_j^l = \begin{cases} 1 & \text{if } \mathbf{x}^j \in Q_l, l = \phi_w(\mathbf{x}^j) \\ 0 & \text{else} \end{cases}$$

où chaque observation \mathbf{x}^j est affectée au plus proche prototype \mathbf{g}^l en utilisant la fonction d'affectation, défini comme suit :

$$\phi_w(\mathbf{x}^j) = \arg \min_l \sum_{i=1}^N \sum_{k=1}^K z_i^k (x_i^j - g_r^l)^2 \quad (3)$$

Le problème 3 est résolu par :

$$g_r^l = \frac{\sum_{k=1}^K \sum_{i=1}^N \sum_{j=1}^d \mathcal{H}^T(\delta(k, r)) z_i^k w_j^l x_i^j}{\sum_{k=1}^K \sum_{i=1}^N \sum_{j=1}^d \mathcal{H}^T(\delta(k, r)) z_i^k w_j^l}$$

Cette valeur est obtenue en résolvant les gradients $\frac{\partial \mathcal{R}}{\partial g_r^l} = 0$

$$g_r^l = \frac{\sum_{k=1}^K \mathcal{H}^T(\delta(k, r)) \sum_{i=1}^N \sum_{j=1}^d z_i^k w_j^l x_i^j}{\sum_{k=1}^K \mathcal{H}^T(\delta(k, r)) \sum_{i=1}^N \sum_{j=1}^d z_i^k w_j^l}$$

$$g_r^l = \frac{\sum_{k=1}^K \sum_{x_i^j \in B_k^l} \mathcal{H}^T(\delta(k, r)) x_i^j}{\sum_{k=1}^K \sum_{x_i^j \in B_k^l} \mathcal{H}^T(\delta(k, r))}$$

On peut réécrire l'équation comme suit :

$$g_r^l = \frac{\sum_{k=1}^K \mathcal{H}^T(\delta(k, r)) \sum_{x_i^j \in B_k^l} x_i^j}{\sum_{k=1}^K \mathcal{H}^T(\delta(k, r)) \sum_{x_i^j \in B_k^l}} \quad (4)$$

Les principales phases de l'algorithme BiTM sont présentées dans l'algorithme 1.

Algorithme 1 : Algorithme BiTM

```

1: Entrées :
   — Matrice de données  $\mathbf{D}$ , prototypes  $\mathbf{G}$  (Initialisation).
   —  $t_f$  : le nombre maximum d'itérations.
2: Sorties :
   — Matrice d'affectation  $\mathbf{Z}$ ,  $\mathbf{W}$ . Prototypes  $\mathbf{G}$ 
3: tantque  $t \leq t_f$  faire
4:   pour tout  $\mathbf{x}_i \in \mathbf{D}$  faire
5:     Phase d'affectation des observations : Chaque observation  $\mathbf{x}_i$  est affectée au prototype le plus proche  $\mathbf{g}_k$  en utilisant la fonction d'affectation, défini dans l'équation 2.
6:     Phase d'affectation des variables : Chaque variable  $\mathbf{x}^j$  est affectée au prototype le plus proche  $\mathbf{g}^l$  en utilisant la fonction d'affectation, définie dans l'équation 3.
7:     Phase quantification : Les vecteurs des prototypes sont mis à jour en utilisant l'expression définie dans l'équation 4.
8:   fin pour
9:   Mise à jour de  $T$ 
   {  $T$  varie de  $T_{max}$  à  $T_{min}$  }
10:  t++
11: fin tantque

```

3.2 BitM et MapReduce

Afin de traiter de gros volumes de données, il est nécessaire d'utiliser des architectures distribuées. Le développement d'algorithmes dans un environnement distribué destiné au traitement de gros volumes de données implique de nombreuses problématiques tel que la répartition de la charge, la localisation des données, la tolérance aux pannes et l'écriture d'algorithmes agnostique au volume des données à traiter. La plateforme Spark gère très bien ces problématiques. Il reste cependant à la charge du développeur d'adapter son algorithme aux contraintes du paradigme MapReduce. Cette étape d'adaptation consiste principalement à décomposer le problème en fonctions élémentaires.

L'algorithme BiTM consiste à itérer sur deux phases de MapReduce, l'une pour traiter les lignes et l'autre pour traiter les colonnes de la matrice de données. À la fin de chaque itération il y a une étape de synchronisation afin de mettre à jour les paramètres \mathbf{G} , \mathbf{W} , \mathbf{Z} . Les fonctions élémentaires définies sont :

- Affectation de chaque observations \mathbf{x}_i au meilleur prototype selon l'expression (Eq. 2).
- Affectation de chaque variable \mathbf{x}^j au meilleur prototype selon l'équation (Eq. 3).
- Somme des dénominateur et numérateur de chaque prototype \mathbf{g}_r (Eq.4)
- Mise à jour des vecteurs des prototype $\mathbf{g}_r, \forall r \in C$ (Eq. 4)

Le pseudo code ci-dessous décrit l'implémentation de l'algorithme BiTM avec Spark MapReduce.

Souvent dans les jeux de données le nombre d'observations est très largement supérieur au nombre de variables ($N \gg d$). Dans ce type de jeux de données, le nombre d'observations peut fréquemment dépasser le million, le vecteur associé à une colonne \mathbf{x}^j est donc de trop grande dimension pour être utilisé directement en entrée de la fonction de calcul de distance.

Ainsi, pour la fonction d'affectation des lignes nous avons défini une fonction spécifique (algorithme 5) qui détermine pour chaque \mathbf{x}_i le prototype le plus proche. L'étape de calcul des prototypes g est séparée en deux fonctions de Reduce (Eq. 4).

Lors de l'affectation des colonnes, la fonction de Map (Algorithme 3), la fonction de reduce (Algorithme 6) ainsi que la fonction d'affectation des colonnes divisent le calcul de la distance d'une colonne en plusieurs sorties ($d \times K$). L'étape du Map calcule la distance entre chaque élément x_i^j avec chaque prototype. Puis l'étape du Reduce réalise la somme de toutes ces distances afin d'obtenir une distance par "colonne-prototype". à la fin de ces étapes les colonnes sont affectées aux prototypes minimisant cette distance (Algorithme 2) et les vecteurs des prototypes sont en suite mis à jour.

Algorithme 2 BiTM - Main

Initialisation

{Initialisation aléatoire des prototypes}

{Affectation aléatoire des colonnes aux prototypes \mathbf{W} }**{Boucle principale}****tantque** $t \leq t_f$ **faire** **{Affectation des colonnes}** **pour tout** $\mathbf{x}_i \in \mathbf{D}$ **faire** $\langle (J, L); V \rangle \leftarrow \text{ColMapper}(\mathbf{x}_i)$ **fin pour** $\langle (J, L); V \rangle \leftarrow \text{ColReducer}(\langle (J, L); V \rangle)$ **pour tout** valeur de ColReducer $j \in J$ **faire** $\phi_w(\mathbf{x}^j) = \arg \min(\langle (j, L); V \rangle)$ **fin pour** **{Affectation des lignes}** **pour tout** $\mathbf{x}_i \in \mathbf{D}$ **faire** $\langle (\mathbf{CM}, \mathbf{CN}) \rangle = \text{RowMapper}(\mathbf{x}_i)$ **fin pour** $(\mathbf{CMs}, \mathbf{CNs}) = \text{RowReducer}(\langle (\mathbf{CM}, \mathbf{CN}) \rangle)$ **{Mise à jour des prototypes}** $\mathbf{G} = \mathbf{CMs}/\mathbf{CNs}$ $t+ = 1$ **fin tantque**

La majorité des fonctions de l'algorithme sont développées en Map et en Reduce donc par conséquent elles sont exécutées en parallèle sur toutes les machines du cluster Spark. C'est pour cette raison que l'algorithme passe très bien à l'échelle.

biclustering

Algorithme 3 ColMapper(\mathbf{x}_i)

pour chaque colonne $j = 1..d$ **faire**
 pour chaque variable du prototype $l = 1..L$ **faire**
 emit $\langle (j, l); (x_i^j - g_r^l)^2 \rangle$
 fin pour
fin pour

Algorithme 4 ColReducer($key(j, l), V$)

$s_v = 0$
pour chaque valeur de map $v \in V$ **faire**
 $s_v += v$
fin pour
emit $\langle (j, l); s_v \rangle$

Algorithme 5 RowMapper(\mathbf{x}_i)

{**MAP distance sur les lignes** : boucle parallélisée sur tous les vecteurs ($\mathbf{x}_i = (x_i^1, x_i^2, \dots, x_i^d)$) }

pour chaque $k = 1..K$ **faire**
 $bmu(k) = \|\mathbf{x}_i - \mathbf{g}_k\|^2$
fin pour
 $\phi_z(\mathbf{x}_i) = \arg \min(\mathbf{bmu})$
{Construction d'un nouveau vecteur compressé : \mathbf{cm} avec $1 \times L$ dimensions}
 $\mathbf{cm}(l) = \sum_{x_i^j \in B_{\phi_z(\mathbf{x}_i)}^l} x_i^j$
 $\mathbf{CM} = \mathbf{h}^{\phi_z} \times \mathbf{cm}$
{Construction d'un nouveau vecteur compressé : \mathbf{cn} avec $1 \times L$ dimensions}
 $\mathbf{cn}(l) = \sum_{\mathbf{x}^j \in B_{\phi_z(\mathbf{x}_i)}^l} 1$
 $\mathbf{CN} = \mathbf{h}^{\phi_z} \times \mathbf{cn}$
{Les matrices \mathbf{CM} et \mathbf{CN} sont donc de taille $K \times L$ }
emit $\langle (\mathbf{CM}, \mathbf{CN}) \rangle$

Algorithme 6 RowReducer($\mathbf{V}(\mathbf{CM}, \mathbf{CN})$)

Initialisation $\mathbf{CMs} \leftarrow 0, \mathbf{CNs} \leftarrow 0$
pour chaque $(\mathbf{CM}, \mathbf{CN}) \in \mathbf{V}$ **faire**
 $\mathbf{CMs} += \mathbf{CM}$
 $\mathbf{CNs} += \mathbf{CN}$
fin pour
emit $\langle (\mathbf{CMs}, \mathbf{CNs}) \rangle$

4 Expérimentations

Les tests de passage à l'échelle de notre algorithme ont été effectués sur le cluster Magi (<http://www.univ-paris13.fr/calcul/wiki/>), en utilisant de 1 à 10 machines. Chaque machine sur ce cluster a 12 coeurs (deux Xeon X5670 à 2.93GHz), 24Go de RAM et sont connectées sur un réseau InfiniBand. Toutes les expérimentations sont réalisées sous la plateforme Spark. Le programme est disponible sur <https://github.com/TugdualSarazin/spark-clustering>.

Les données sont générées aléatoirement selon la loi normal, à chaque valeur est appliqué un facteur (eq. $x = a + \mathcal{N} * b$) selon la classe à laquelle elle est affectée. Les coefficients a et b sont différents pour chaque classe.

Pour l'analyse des performances de notre implémentation de BITM MapReduce, nous avons généré de 1 à 2 millions d'observations et de 10 à 40 dimensions. Nous avons utilisé une carte BiTM de 10×10 en faisant varier le nombre de coeurs à chaque exécution. Les temps sont indiqués en millisecondes. Dans toutes les figures nous avons ajouté une courbe correspondant au temps idéal. Elle est déterminé à partir du temps d'exécution obtenu sur une machine divisé par le temps d'execution du nombre de machines utilisé par la suite. Cette mesure est définie comme suit :

$$\text{speedup } n = \text{temps d'execution utilisant 1 coeur} / \text{temps d'execution utilisant } n \text{ coeurs}$$

Dans la figure 1(a) le temps en pratique est très largement supérieur au temps idéal. Ceci peut s'expliquer par la faible taille de la matrice de données (1 million de valeurs), le système utilise alors beaucoup de temps de calcul pour répartir les calculs et transférer les données entre toutes les machines. Lorsque le nombre de données augmente ce temps de transfert et de répartition devient insignifiant comme on peut le constater dans les figures 1(b), 1(c) et 1(d). Dans la figure 1(e) le temps d'exécution en pratique est parfois inférieur au temps idéal. Ceci peut s'expliquer par une saturation des ressources du système lorsqu'il n'y a qu'une seule machine dans le système.

Ces tests prouvent que notre implémentation sur Spark passe très bien à l'échelle en approchant régulièrement des temps d'exécution proche du temps idéal.

5 Conclusion

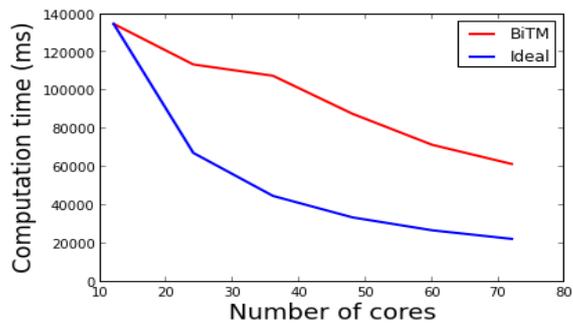
Dans cet article, nous avons proposé une nouvelle approche de biclustering topologique basée sur les cartes d'auto-organisatrices. Nous avons précédemment montré son efficacité en terme de qualité de clustering. Dans ces travaux nous avons montré ses capacités à passer à l'échelle afin de traiter de gros volumes de données via l'implémentation sous la plateforme Spark et le paradigme MapReduce.

Ces résultats préliminaires montrent que les méthodes de conception de BiTM peuvent être étendues à d'autres algorithmes de biclustering. C'est pourquoi nous avons réalisé une bibliothèque Open source d'algorithmes de clustering et de biclustering basé sur Spark.

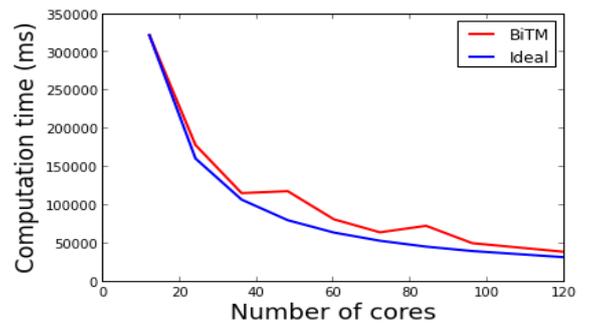
Dans nos futurs travaux, nous prévoyons de tester nos algorithmes sur de grands volumes de données, particulièrement sur les données réelles.

Remerciements : Ce travail a été soutenu et réalisé dans le cadre du projet "Square Predict" (investissement d'avenir – Big data) et du contrat CIFRE-ANRT.

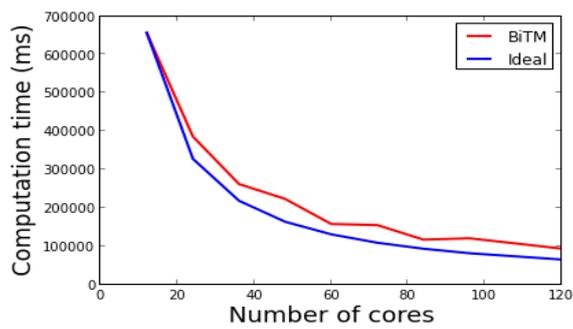
biclustering



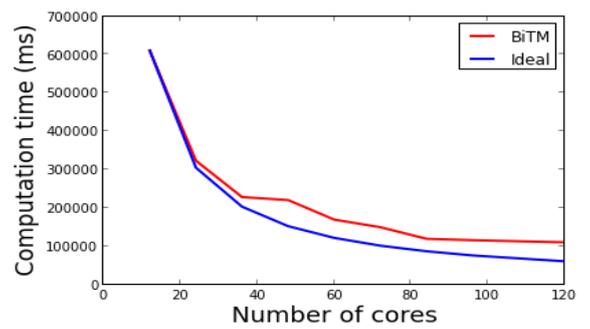
(a) 1 million observations, 10 variables



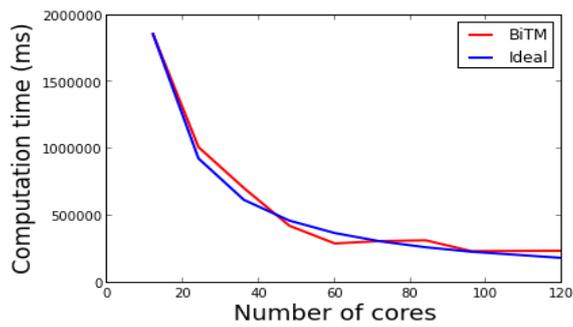
(b) 1 million observations, 20 variables



(c) 1 million observations, 40 variables



(d) 2 millions observations, 20 variables



(e) 2 millions observations, 40 variables

FIG. 1 – Temps d'exécution de BiTM Spark

Références

- Chaibi, A., H. Azzag, et M. Lebbah (2014). Pondération de blocs de variables en bipartitionnement topologique. In *14èmes Journées Francophones Extraction et Gestion des Connaissances, EGC 2014, Rennes, France, 28-32 Janvier, 2014*, pp. 317–328.
- Dean, J. et S. Ghemawat (2008). Mapreduce : simplified data processing on large clusters. *Commun. ACM* 51(1), 107–113.
- Ene, A., S. Im, et B. Moseley (2011). Fast clustering using mapreduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, New York, NY, USA*, pp. 681–689. ACM.
- Ferreira Cordeiro, R. L., C. Traina, Junior, A. J. Machado Traina, J. López, U. Kang, et C. Faloutsos (2011). Clustering very large multi-dimensional datasets with mapreduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, New York, NY, USA*, pp. 690–698. ACM.
- George, T. et S. Merugu (2005). A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining, ICDM '05, Washington, DC, USA*, pp. 625–628. IEEE Computer Society.
- Ghoting, A., P. Kambadur, E. Pednault, et R. Kannan (2011). Nimble : a toolkit for the implementation of parallel data mining and machine learning algorithms on mapreduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, New York, NY, USA*, pp. 334–342. ACM.
- Govaert, G. (1983). *Classification croisée*. Ph. D. thesis, Université Paris 6, France.
- Govaert, G. et M. Nadif (2008). Block clustering with Bernoulli mixture models : Comparison of different approaches. *Computational Statistics and Data Analysis* 52, 3233–3245.
- Karloff, H. J., S. Suri, et S. Vassilvitskii (2010). A model of computation for mapreduce. In *SODA '10*, pp. 938–948.
- Kohonen, T. (2001). *Self-organizing Maps*. Springer Berlin.
- Labioud, L. et M. Nadif (2011). Co-clustering under nonnegative matrix tri-factorization. In *Proceedings of the 18th international conference on Neural Information Processing - Volume Part II, ICONIP' 11, Berlin, Heidelberg*, pp. 709–717. Springer-Verlag.
- Lee, D. D. et H. S. Seung (1999). Learning the parts of objects by nonnegative matrix factorization. *Nature* 401, 788–791.
- Lin, C., Y. Yang, et T. Rutayisire (2011). A parallel cop-kmeans clustering algorithm based on mapreduce framework. In Y. Wang et T. Li (Eds.), *Knowledge Engineering and Management*, Volume 123 of *Advances in Intelligent and Soft Computing*, pp. 93–102. Springer Berlin Heidelberg.
- Liu, W. et L. Chen (2008). A parallel algorithm for gene expressing data biclustering. *JCP* 3(10), 71–77.
- Long, B., Z. M. Zhang, et P. S. Yu (2005). Co-clustering by block value decomposition. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, KDD '05, New York, NY, USA*, pp. 635–640. ACM.
- Low, Y., D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, et J. M. Hellerstein (2012). Dis-

biclustering

- tributed graphlab : A framework for machine learning and data mining in the cloud. *Proc. VLDB Endow.* 5(8), 716–727.
- Lv, Z., Y. Hu, H. Zhong, J. Wu, B. Li, et H. Zhao (2010). Parallel k-means clustering of remote sensing images based on mapreduce. In *Proceedings of the 2010 international conference on Web information systems and mining, WISM'10*, Berlin, Heidelberg, pp. 162–170. Springer-Verlag.
- Malewicz, G., M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, et G. Czajkowski (2010). Pregel : A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10*, New York, NY, USA, pp. 135–146. ACM.
- Papadimitriou, S. et J. Sun (2008). Disco : Distributed co-clustering with map-reduce : A case study towards petabyte-scale end-to-end mining. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pp. 512–521. IEEE.
- Priam, R., M. Nadif, et G. Govaert (2008). The block generative topographic mapping. In L. Prevost, S. Marinai, et F. Schwenker (Eds.), *The Third International Workshop on Artificial Neural Networks in Pattern Recognition, Lecture Notes in Artificial Intelligence (LNCS)*, Volume 5064 of *Lecture Notes in Computer Science*, pp. 13–23. Springer.
- Sul, S.-J. et A. Tovchigrechko (2011). Parallelizing blast and som algorithms with mapreduce-mpi library. In *IPDPS Workshops'11*, pp. 481–489.
- Zaharia, M., M. Chowdhury, M. J. Franklin, S. Shenker, et I. Stoica (2010). Spark : cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, HotCloud'10*, Berkeley, CA, USA, pp. 10–10. USENIX Association.

Summary

Biclustering is a main task in a variety of areas of machine learning and data mining providing simultaneous observations and features clustering. Biclustering approaches are more complex compared to the traditional clustering particularly those requiring large dataset and Mapreduce platforms. We propose a new approach of biclustering based on popular self-organizing maps, which is one of the famous unsupervised learning algorithms for cluster analysis of large dataset. We have designed scalable implementations of the new biclustering algorithm BiTM (Biclustering Topological Map) using MapReduce with the Spark platform. We report the experiments and demonstrated the performance in terms of speedup with public dataset using different cores. Using practical examples, we demonstrate that our algorithm works well in practice. The experimental results show scalable performance with near linear speedups across different data and 120 cores.

Clustering conceptuel à base de motifs localement optimaux

Frédéric Pennerath*

*École Centrale Supélec, 2 rue Edouard Belin, 57000 Metz
frederic.pennerath@centralesupelec.fr,
<http://metz.supelec.fr/metz/personnel/pennerath>

Résumé. L'extraction des motifs localement optimaux est une méthode de clustering conceptuel adaptée aux données symboliques. Contrairement aux autres méthodes de clustering à base de motifs, les motifs localement optimaux s'obtiennent en éliminant les motifs similaires plutôt qu'en éliminant les motifs qui couvrent les mêmes données. Si cette approche correspond mieux à la façon dont un expert analyse les données, la couverture des données par les motifs apporte un complément d'information important. L'objet de cet article est de mettre en rapport ces deux formes de redondance dans l'espace des données et dans l'espace des motifs, puis de montrer de quelle manière le modèle des motifs localement optimaux peut les intégrer dans son processus de sélection pour produire un clustering hiérarchique de granularité modulable.

La plupart des techniques actuelles de clustering traitent du problème du regroupement d'échantillons plongés dans un espace vectoriel de grande dimension. Même lorsque les données sont essentiellement de nature symbolique, comme des données textuelles ou relationnelles, il n'est pas rare de les projeter artificiellement dans un tel espace pour pouvoir leur appliquer des méthodes dites numériques. Pourtant dans les années 80, des méthodes dites de *clustering conceptuel* comme CobWeb (Fisher, 1987) ont été conçues spécialement pour la classification non supervisée de données symboliques. Le regroupement de données en clusters ne repose alors plus sur une notion de distance mais sur un prédicat logique plus ou moins complexe que les données d'un même cluster satisfont. Les clusters sont appelés *concepts* et chaque concept est représenté à la fois par son *intension* (i.e. la condition définie dans l'espace de description des données) et par son *extension* (i.e. l'ensemble des données vérifiant la condition). Le clustering conceptuel est particulièrement adapté à l'analyse de données, puisque l'intension de chaque concept suggère immédiatement une généralisation possible de ce cluster. L'apparition dans les années 90 de la *recherche de motifs fréquents* (Agrawal et Srikant, 1994) a permis de reconsidérer le clustering conceptuel sous un angle légèrement différent. Étant donné un ensemble \mathcal{M} de motifs muni d'une relation d'ordre $\subseteq_{\mathcal{M}}$, une donnée est décrite par un motif M si M est contenu (au sens de $\subseteq_{\mathcal{M}}$) dans la description de la donnée exprimée dans \mathcal{M} . Dans le cas de données décrites par un ensemble d'attributs monovalués \mathcal{A} , l'espace des motifs est l'ensemble $2^{\mathcal{A}}$ des sous-ensembles d'attributs ordonnés par l'inclusion ensembliste. Mais l'espace des motifs peut être plus complexe, comme par exemple l'ensemble des graphes connexes non isomorphes deux à deux ordonnés par la relation de sous-graphe isomorphe. La *fréquence* $\sigma(M)$ d'un motif M est le nombre de données décrites par celui-ci. Les algorithmes de recherche de motifs fréquents permettent de calculer la fréquence de tous les motifs *fréquents*, c'est-à-dire des motifs dont la fréquence est supérieure à un seuil fixé arbitrairement.

Depuis les années 2000, différentes méthodes ont été proposées pour sélectionner parmi les nombreux motifs ainsi obtenus un sous-ensemble restreint de motifs représentatifs des données (Knobbe et Ho, 2006; Pennerath et Napoli, 2009; Mampaey et al., 2011; Van Leeuwen et Knobbe, 2011). Les motifs retenus sont interprétables comme les intensions d'un ensemble de concepts, l'extension $\text{ext}(M)$ d'un motif M étant l'ensemble des données dont l'intension contient M . Si la justification théorique de la caractérisation des motifs à extraire varie d'une méthode à l'autre, toutes ces méthodes visent à produire des motifs à la fois peu nombreux, représentatifs des données (c'est-à-dire de fréquence relativement élevée) et porteurs d'une information non redondante. C'est essentiellement sur la façon de définir et quantifier cette redondance que des divergences apparaissent entre méthodes, en particulier selon que la redondance se définit dans l'espace des données ou dans l'espace des motifs. L'objet de cet article est premièrement d'étudier ces deux formes de redondance en mettant en lumière leurs différences et leur complémentarité, et deuxièmement de montrer comment le modèle des motifs localement optimaux (Pennerath et Napoli, 2009) peut être modifié pour tenir compte simultanément des deux formes de redondance et ainsi aboutir à un clustering hiérarchique dont le niveau de granularité est modulable.

La plupart des méthodes de clustering à base de motifs (Knobbe et Ho, 2006; Mampaey et al., 2011; Van Leeuwen et Knobbe, 2011) considèrent que deux motifs sont redondants s'ils décrivent approximativement les mêmes données, c'est-à-dire si leurs extensions se recouvrent fortement. Ce type de méthode tend à produire des clusters de données approximativement disjoints, souvent de tailles comparables. Cette forme de redondance a l'avantage de pouvoir être estimée rapidement en mesurant le degré de superposition des extensions, par exemple à l'aide de l'indice de Jaccard. Elle présente l'inconvénient majeur de ne pas tenir compte de ce que les motifs expriment. La procédure de sélection est aveugle et peut éliminer parmi deux motifs redondants celui qui contenait l'information cruciale pour l'expert. Plutôt que de quantifier la redondance de motifs dans l'espace des données, le modèle des *motifs localement optimaux* (Pennerath et Napoli, 2009) mesure la redondance directement dans l'espace des intensions, c'est-à-dire dans \mathcal{M} : deux motifs seront redondants s'ils sont proches dans l'espace des motifs au sens de la distance d'édition. Ce type de méthode induit des clusters caractérisés par des motifs généralement bien différenciés. L'avantage de cette forme de redondance est d'agir directement dans l'espace de description que perçoit l'expert pour conduire son analyse. Son principal inconvénient est de nécessiter des calculs plus lourds, en particulier lorsque les motifs sont complexes comme les graphes. Les deux formes de redondance sont fortement liées : deux motifs similaires auront tendance à décrire les mêmes sous-ensembles de données, et réciproquement. Il existe toutefois des cas où une légère modification de l'expression d'un motif entraîne une modification radicale de son extension. Ce phénomène peut être révélateur de la frontière entre nouveaux concepts et devrait être pris en compte à ce titre.

De ce fait, le modèle des motifs localement optimaux, qui sélectionne déjà les concepts sur la base de leurs intensions, a été modifié pour intégrer aussi la comparaison de leurs extensions. Ce modèle considère un espace des motifs \mathcal{M} ordonné par une relation d'inclusion $\subseteq_{\mathcal{M}}$ et une fonction de score $s : (\mathcal{M}, \subseteq_{\mathcal{M}}) \rightarrow (\mathbb{S}, \leq_{\mathbb{S}})$ associant à chaque motif M un score $s(M)$ prenant sa valeur dans l'ensemble \mathbb{S} muni d'une relation d'ordre $\leq_{\mathbb{S}}$. Le choix de la fonction de score est laissé à l'expert en charge de l'analyse des données. En pratique cette fonction est généralement choisie de manière à privilégier l'extraction de concepts qui présentent simultanément une intension et une extension informatives. Par exemple, la fonction d'aire s_a qui associe à

un motif M son aire, c'est-à-dire le produit $s_a(M) = |M| \cdot \sigma(M)$ de la longueur du motif $|M|$ par sa fréquence $\sigma(M)$, exprime un compromis entre la longueur du motif et sa fréquence. Les motifs localement optimaux sont ceux qui maximisent localement la fonction de score au sein de l'ordre des motifs. Plus formellement, deux motifs sont dits *voisins* si l'un est le *successeur immédiat* de l'autre, c'est à dire s'ils sont comparables selon $\subseteq_{\mathcal{M}}$ mais qu'aucun autre motif intermédiaire n'existe dans l'intervalle de comparaison que ces deux motifs définissent. Un motif M *domine* un motif M' si M et M' sont voisins et si le score de M est strictement supérieur à celui de M' selon \leq_s . Un motif est *localement optimal* s'il n'est dominé par aucun de ses voisins. Cette définition équivaut à une procédure de présélection qu'un expert pourrait faire rationnellement à partir d'un ensemble de motifs munis de leur fréquence : d'abord estimer l'intérêt de chaque motif par un score tenant compte de sa fréquence mais aussi d'autres paramètres intrinsèques au motif ; ensuite trier par ordre décroissant de scores les motifs ; enfin parcourir la liste triée et n'étudier que les motifs qui ne sont pas similaires à des motifs de score plus élevé. Cette procédure correspond ni plus ni moins à l'extraction des motifs localement optimaux et a pu être automatisée à l'aide d'algorithmes optimisés (Pennerath, 2010).

La sélection des motifs ne repose donc pas directement sur la comparaison des extensions de motifs. Si par exemple le motif M_2 est un successeur immédiat du motif M_1 , M_2 sera désavantagé par rapport à M_1 du fait d'une fréquence plus faible mais avantagé par une information structurelle plus riche. La fonction de score a donc pour rôle d'arbitrer entre ces deux phénomènes et d'établir qui des deux motifs domine l'autre. La relation de dominance sert ainsi à sélectionner le motif localement optimal qui émerge parmi un grand nombre de motifs semblables reliés par des variations insignifiantes de score. Il peut toutefois arriver que des variations importantes de score apparaissent lors du passage du motif M_1 au motif M_2 , du fait d'un décrochement important de la fréquence. Dans ce cas l'élimination de M_1 ou de M_2 qui en résulte n'est plus justifiée car la chute brutale de score exprime un véritable contraste entre motifs dont on veut garder une trace. En d'autres termes, on aimerait que deux motifs soient redondants non seulement si leurs intensions (les motifs) sont similaires mais aussi si leurs extensions sont semblables. On ajoute donc une condition supplémentaire pour qu'un motif M_1 domine un motif voisin M_2 : il faut en plus que leur indice de Jaccard $J(\text{ext}(M_1), \text{ext}(M_2)) = \frac{|\text{ext}(M_1) \cap \text{ext}(M_2)|}{|\text{ext}(M_1) \cup \text{ext}(M_2)|}$ soit supérieur à un seuil δ fixé compris entre 0 et 1. Heureusement l'ajout de cette condition n'entraîne presque aucun calcul supplémentaire. En effet si M_2 est le successeur immédiat de M_1 , alors $\text{ext}(M_2) \subseteq \text{ext}(M_1)$ et donc $J(\text{ext}(M_1), \text{ext}(M_2)) = \frac{|\text{ext}(M_2)|}{|\text{ext}(M_1)|} = \frac{\sigma(M_2)}{\sigma(M_1)}$. Le calcul de la fréquence des motifs étant déjà nécessaire pour établir le score des motifs, le calcul du rapport des fréquences est immédiat.

L'algorithme d'extraction des motifs localement optimaux décrit dans Pennerath (2010) a pu être modifié pour tenir compte de cette nouvelle condition et des tests ont été conduits pour étudier l'influence du paramètre δ sur des jeux de test classiques. Quand $\delta = 0$, on retrouve évidemment les motifs localement optimaux et quand $\delta = 1$, on obtient les *motifs fermés* (Bastide et al., 2000) beaucoup trop nombreux. Entre ces deux extrêmes, on voit apparaître une hiérarchie de clusters intermédiaires dont le nombre augmente avec δ . Il est intéressant de constater que d'une part cette hiérarchie de concepts est stable (i.e. un concept pour une valeur δ_1 sera aussi concept pour toute valeur δ_2 supérieure à δ_1) et d'autre part cette hiérarchie se développe de façon équilibrée, sans devenir horizontale (i.e avec un grand nombre de concepts indépendants) ni verticale (avec une suite de concepts inclus les uns dans les autres). D'autres tests ont pu montrer que le modèle s'adapte aussi aux problèmes de classification supervisée

en utilisant des fonctions de score favorisant les motifs à fort contraste (très présents dans une classe et peu dans les autres). En conclusion le modèle des motifs localement optimaux permet de prendre en compte simultanément la redondance des intensions et des extensions des concepts extraits afin de produire une hiérarchie de concepts présentant un niveau de détails facilement ajustable.

Références

- Agrawal, R. et R. Srikant (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases, (VLDB'94), Santiago de Chile, Chile*, pp. 478–499. Morgan Kaufmann.
- Bastide, Y., R. Taouil, N. Pasquier, G. Stumme, et L. Lakhal (2000). Mining frequent patterns with counting inference. *ACM SIGKDD Explorations* 2(2), 66–75.
- Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine learning* 2(2), 139–172.
- Knobbe, A. J. et E. K. Ho (2006). Pattern teams. In *Knowledge Discovery in Databases : PKDD 2006*, pp. 577–584. Springer.
- Mampaey, M., N. Tatti, et J. Vreeken (2011). Tell me what i need to know : succinctly summarizing data with itemsets. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 573–581. ACM.
- Pennerath, F. (2010). Fast extraction of locally optimal patterns based on consistent pattern function variations. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010*, Volume 6323, pp. 34–49. Springer.
- Pennerath, F. et A. Napoli (2009). The model of most informative patterns and its application to knowledge extraction from graph databases. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009*, Volume 5782, pp. 205–220. Springer.
- Van Leeuwen, M. et A. Knobbe (2011). Non-redundant subgroup discovery in large and complex data. In *Machine Learning and Knowledge Discovery in Databases*, pp. 459–474. Springer.

Summary

The extraction of locally optimal patterns is a method for clustering symbolic data. Compared to other clustering methods based on patterns, locally optimal patterns are extracted by removing similar patterns rather than removing patterns covering the same subset of data. Whereas this approach better corresponds to the way experts analyse data, the way patterns cover data is also an element of information that should be taken into account. The goal of this article is to relate both forms of redundancy in the space of data and in the space of patterns, then to show how the model of locally optimal patterns can integrate them in order to produce a hierarchical clustering with a scalable granularity.

Analyse exploratoire par k -Cocustering avec Khiops CoViz

Bruno Guerraz*, Marc Boullé*, Dominique Gay*,
Vincent Lemaire*, Fabrice Clérot*

*Orange Labs

2, avenue Pierre Marzin, F-22307 Lannion Cedex, France
firstname.name@orange.com

Résumé. En analyse exploratoire, l'identification et la visualisation des interactions entre variables dans les grandes bases de données est un défi (Dhillon et al., 2003; Kolda et Sun, 2008). Nous présentons Khiops CoViz, un outil qui permet d'explorer par visualisation les relations importantes entre deux (ou plusieurs) variables, qu'elles soient catégorielles et/ou numériques. La visualisation d'un résultat de cocustering de variables prend la forme d'une grille (ou matrice) dont les dimensions sont partitionnées: les variables catégorielles sont partitionnées en clusters et les variables numériques en intervalles. L'outil permet plusieurs variantes de visualisations à différentes échelles de la grille au moyen de plusieurs critères d'intérêt révélant diverses facettes des relations entre les variables.

1 Khiops CoViz : Visualisation des modèles en grille

Khiops CoViz, développée en Flex, est la brique logicielle de visualisation de Khiops Cocustering (KHC)¹. Étant données, deux (ou plus) variables catégorielles ou numériques, KHC réalise un partitionnement simultané des variables : les valeurs de variables catégorielles sont groupés en clusters et les variables numériques sont partitionnées en intervalles – ce qui revient à un problème de cocustering. Le produit des partitions uni-variées forme une partition multivariée de l'espace de représentation, i.e., une grille ou matrice de cellules et il représente aussi un estimateur de densité jointe des variables. Afin de choisir la “meilleure” grille M^* (connaissant les données) de l'espace de modèles \mathcal{M} , nous exploitons une approche Bayésienne dite Maximum A Posteriori (MAP). KHC explore l'espace de modèles en minimisant un critère Bayésien, appelé $cost$, qui réalise un compromis entre la précision et la robustesse du modèle :

$$cost(M) = -\log(\underbrace{p(M | D)}_{\text{posterior}}) = -\log(\underbrace{p(M)}_{\text{prior}} \times \underbrace{p(D | M)}_{\text{vraisemblance}}) \quad (1)$$

KHC construit aussi une hiérarchie des parties de chaque dimensions (i.e., clusters ou intervalles adjacents) en utilisant une stratégie agglomérative ascendante, en partant de M^* , la grille optimale résultant de la procédure d'optimisation, jusqu'à M_\emptyset , le modèle nul, i.e., la grille (unicellulaire) où aucune dimension n'est partitionnée. Les hiérarchies sont construites en fusionnant les parties qui minimisent l'indice de dissimilarité $\Delta(c_1, c_2) = cost(M_{c_1 \cup c_2}) - cost(M)$,

1. <http://www.khiops.com> – Pour plus de détails sur l'implémentation de KHC, voir Boullé

où c_1, c_2 sont deux parties d'une partition d'une dimension de la grille M et $M_{c_1 \cup c_2}$ la grille après fusion de c_1 et c_2 . De cette manière, la fusion de parties minimise la dégradation du critère $cost$, donc minimise la perte d'information par rapport à la grille M avant fusion. L'utilisateur peut ainsi choisir la granularité de la grille nécessaire à son analyse tout en contrôlant soit le nombre de parties soit le taux d'information (i.e., le pourcentage d'information gardé dans le modèle : $IR(M') = (cost(M') - cost(\mathcal{M}_0)) / (cost(M^*) - cost(\mathcal{M}_0))$). La grille optimale M^* et les hiérarchies correspondantes constituent les principales structures de notre outil de visualisation.

2 Interface utilisateur : Exploration & Interactivité

La figure 1 présente l'interface utilisateur de l'outil pour un sous-ensemble de données de la base DBLP. Nous considérons des données à trois dimensions ($Author \times Year \times Event$) pour 16 000 auteurs ayant publié au moins une fois dans une des principales conférences de Bases de Données ou de Fouille de Données – $Year$ est l'année de la publication. Le panneau principal de visualisation de l'outil est composé des hiérarchies de deux dimensions sélectionnées, des parties terminales des hiérarchies et de la composition d'une partie sélectionnée. La visualisation de la grille correspondante est aussi disponible via le panneau principal (voir la figure 2 à gauche). L'outil permet de naviguer parmi les parties d'une dimension tandis que les deux autres dimensions sont fixées et dédiées à la visualisation (pour le cas à plus de deux variables).

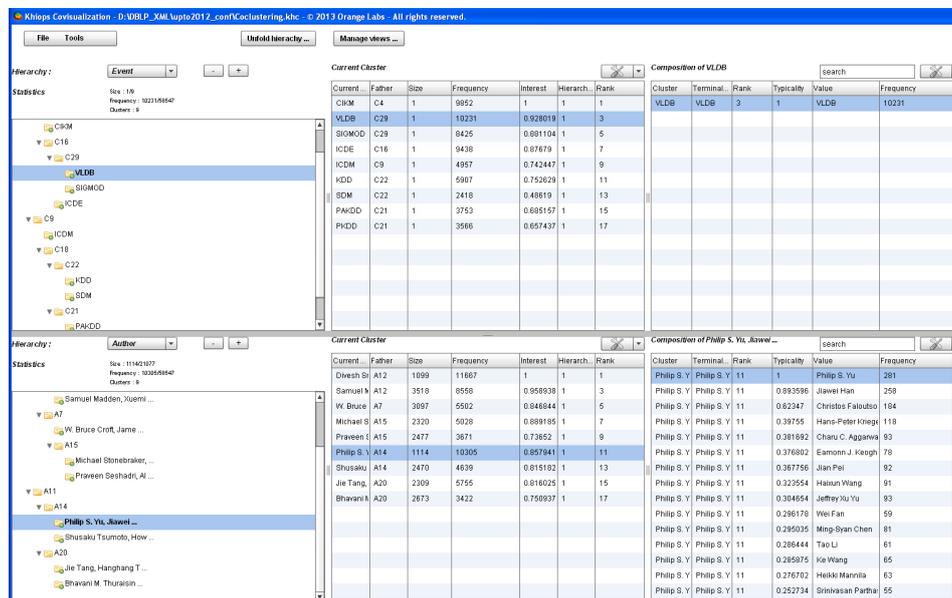


FIG. 1 – Panneau principal de Khiops CoViz : (de gauche à droite), les hiérarchies des parties de deux dimensions ($Author$ and $Event$), la liste des parties terminales des hiérarchies et la composition des parties sélectionnées.

Choisir la granularité voulue pour visualiser la grille se fait au moyen de la fonctionnalité “unfold hierarchy” (voir figure 2 à droite). Un analyste peut contrôler le nombre de parties par dimension ou le taux d’information (comme décrit plus haut) par fusion optimale ou par fusion personnalisée (non-optimale). Selon les données d’applications, la grille optimale peut être composée de beaucoup de clusters (peut-être trop pour l’analyste) et fusionner des clusters augmente mécaniquement le nombre de valeurs par cluster : pour faciliter l’analyse des clusters, l’outil propose deux mesures, l’*intérêt* d’un cluster et la *typicité* d’une valeur dans un cluster. Ces deux mesures sont dérivées du critère *cost* (Guigourès, 2013) et sont utiles pour ordonner les clusters ou les valeurs de clusters par intérêt et se focaliser sur les composantes les plus intéressantes (voir figure 1).

Panneau de visualisation. Pour deux variables partitionnées X et Y à visualiser, les visualisations classiques, telles les “heat map” sont disponibles : il est ainsi possible de visualiser la fréquence des cellules, la probabilité jointe, la probabilité conditionnelle, la densité jointe ou conditionnelle. De plus, l’outil propose deux critères (dérivés de l’information mutuelle $MI(X, Y)$, voir Guigourès (2013) pour les définitions complètes) qui fournissent des informations supplémentaires sur les interactions entre variables :

- Contribution à l’information mutuelle (CMI) : CMI indique comment les cellules contribuent à l’information mutuelle $MI(X, Y)$; positivement (rouge), négativement (bleu), nullement (blanc) indique respectivement un excès d’interactions, un déficit ou aucune interaction particulière comparée à ce qui est attendu en cas d’indépendance des variables partitionnées.
- Contraste : Le contraste est dédié aux grilles 3D (ou plus). Étant données deux variables partitionnées fixées X, Y à visualiser et une partie P_i d’une troisième variable K , appelée contexte, le contraste met en lumière les cellules qui caractérisent P_i par rapport à toute les données. Comme précédemment, un contraste positif, négatif ou nul est différencié par les couleurs rouge, bleue et blanche.

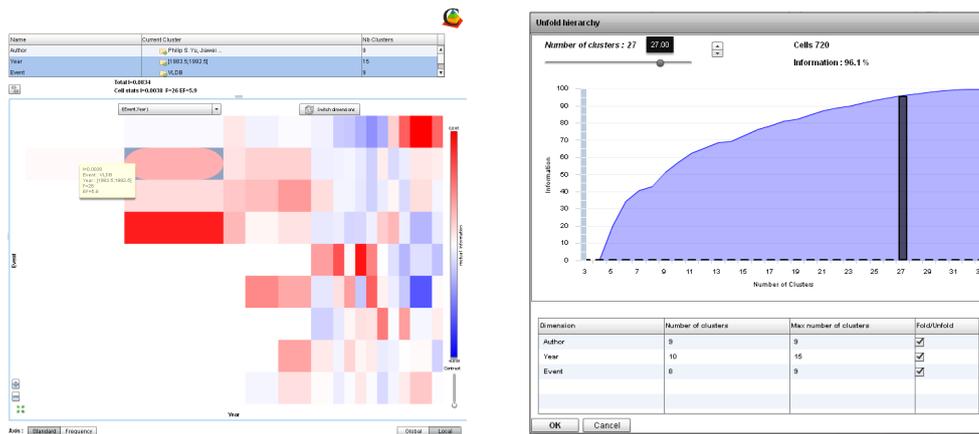


FIG. 2 – (Gauche) : Visualisation de la contribution à l’information mutuelle pour Year \times Event. (Droite) : Panneau de sélection de la granularité de la grille.

3 Usage multiple des modèles en grille

L'analyse exploratoire des interactions entre variables par le biais de la visualisation des modèles en grilles est adaptée à plusieurs types de données et domaines d'applications (Bondu et al., 2013). Nous présentons une liste non-exhaustive des données d'applications typiques qui ont déjà été étudié via Khiops CoViz :

- Marketing : Les clients avec la liste des produits achetés ($customer \times product$)
- Web Mining : analyse de logs web pour identifier des comportements de navigation ($cookies \times webpages$)
- Télécommunications : Dimensionnement de réseau mobile par l'analyse des compte-rendus d'appels (CDRs) ($sourceAntenna \times targetAntenna$), e.g., analyse exploratoire des CDRs à l'échelle d'un pays (Guigourès, 2013).
- Fouille de textes : (co)clustering de textes ($Texts \times Words$)
- Fouille de graphes : Données multigraphes temporels ($SourceNodes \times TargetNodes \times Time$), e.g., analyse des locations de vélos à Londres (Guigourès et al., 2012).
- Clustering de données fonctionnelles (séries temporelles numériques ou catégorielles) : $TimeSeriesId \times Time \times Value$ ou $TimeSeriesId \times Time \times Event$, e.g., clustering de courbes (Boullé, 2012) ou analyse de consommation électrique (Boullé et al., 2012).

4 Conclusion & Travaux futurs

Nous présentons Khiops CoViz, un outil basé sur les modèles en grilles, pour identifier et visualiser les interactions intéressantes entre variables catégorielles et/ou numériques. Cependant, pour de nombreuses applications, l'analyste a souvent besoin de plus qu'une représentation matricielle des résultats, e.g., de représentations graphiques pour les données de graphes, des projections sur cartes pour les données géographiques : des extensions de l'outil par des plug-ins dédiés aux applications sont actuellement étudiées.

Références

- Bondu, A., M. Boullé, et D. Gay (2013). Les modèles en grilles. Principes, évaluation, algorithmes et applications. *Tutorial given at EGC*.
- Boullé, M. Data grid models for preparation and modeling in supervised learning. In *Hands-On Pattern Recognition : Challenges in Machine Learning, volume 1*.
- Boullé, M. (2012). Functional data clustering via piecewise constant nonparametric density estimation. *Pattern Recognition 45*(12), 4389–4401.
- Boullé, M., R. Guigourès, et F. Rossi (2012). Nonparametric hierarchical clustering of functional data. In *EGC (best of volume)*, pp. 15–35.
- Dhillon, I. S., S. Mallela, et D. S. Modha (2003). Information-theoretic co-clustering. In *KDD'03*, pp. 89–98. ACM Press.
- Guigourès, R. (2013). *Utilisation des modèles de co-clustering pour l'analyse exploratoire des données*. Ph. D. thesis, Université Paris 1 Panthéon-Sorbonne.
- Guigourès, R., M. Boullé, et F. Rossi (2012). A triclustering approach for time evolving graphs. In *ICDM Workshops*, pp. 115–122.
- Kolda, T. G. et J. Sun (2008). Scalable tensor decompositions for multi-aspect data mining. In *IEEE ICDM'08*, pp. 363–372.

Index

A

Azzag, Hanane.....5

B

Boullé, Marc.....21

C

Clérot, Fabrice 21

G

Gay, Dominique 21

Guerraz, Bruno21

L

Lebbah, Mustapha.....5

Lemaire, Vincent 21

P

Pennerath, Frédéric.....17

S

Sarazin, Tugdual.....5

V

Vrain, Christel..... 1

