Bipartitionnement d'un tableau de contingence

Malika Charrad[*,**], Yves Lechevallier[***]
Mohamed Ben Ahmed[*]

[*]École Nationale des Sciences de l'Informatique
[**]Conservatoire National des Arts et Métiers
malika.charrad@riadi.rnu.tn,
[***]INRIA-Rocquencourt
Yves.lechevallier@inria.fr

Résumé. La recherche simultanée de partitions sur l'ensemble de lignes et l'ensemble de colonnes d'un tableau de données a donné naissance à des méthodes de classification simultanée ou bipartitionnement. On parle aussi de la classification croisée ou la classification par blocs. Plusieurs algorithmes de bipartitionnement ont été proposés dans la littérature selon le type de tableau des données. Nous nous intéressons dans ce papier à l'algorithme Croki2 de classification croisée des tableaux de contingence. Nous proposons dans ce papier une variante plus rapide de cet algorithme que nous comparons à la version originale à travers des expérimentations sur des données présentant une structure de biclasses générées artificiellement selon une méthodologie que nous détaillons.

1 Introduction

Les méthodes de classification automatique appliquées à des tableaux mettant en jeu deux ensembles de données agissent de façon dissymétrique en ne faisant porter la structure recherchée que sur un seul ensemble. L'application d'une classification sur chaque ensemble est possible mais la détermination des liens entre les deux partitions est difficile. La recherche de structures de classes symétriques, plus précisément, la recherche simultanée de partitions sur les deux ensembles a donné naissance à des méthodes de classification simultanée ou de bipartitionnement. On parle aussi de la classification croisée ou la classification par blocs. Ce type de classification est connu dans la littérature anglaise sous différents noms. Souvent on parle de "two-mode clustering", "two-side clustering", "two-way clustering", "direct clustering" (Hartigan, 1972), "biclustering" (Mirkin, 1996) ou encore "co-clustering" (Dhillon et al., 2003). Ces approches diffèrent souvent dans les algorithmes employés, la nature des blocs recherchés qui peuvent être isolés ou imbriqués, le nombre de blocs identifiés dans les données et la nécessité de fixer le nombre de classes sur les lignes et les colonnes. Ce type d'approches a suscité beaucoup d'intérêt dans divers domaines, en particulier celui des biopuces où l'objectif est de caractériser des groupes de gènes par des groupes de conditions expérimentales. Cependant, les travaux de synthèse sur les algorithmes de bipartitionnement sont concentrés sur les algorithmes appliqués en bioinformatique tels que les travaux de Madeira et Oliveira (2004) et Tanay et al. (2004). Par ailleurs, les algorithmes de classification directe (ou block

clustering) proposés par Hartigan (1975) et les travaux de Govaert (1983) sur la classification croisée trouvent aujourd'hui leur application en *web usage mining*, recherche d'information et *text mining*. D'autres algorithmes sont également proposés pour la classification simultanée tels que les algorithmes basés sur le modèle de mélange (Nadif et Govaert, 2005) (Govaert et Nadif, 2009) et ceux basés sur la théorie de l'information (Dhillon et al., 2003) (Robardet, 2002).
Nous nous concentrons dans ce papier sur les algorithmes de bipartitionnement des tableaux de contingence, en particulier l'algorithme Croki2. Nous présentons alors dans la section suivante un état de l'art sur les algorithmes et les approches de bipartitionnement. La troisième section est consacrée à la présentation en détail de l'algorithme Croki2. Notre contribution fait l'objet de la quatrième section.

2 État de l'art sur le bipartitionnement

2.1 Principe général de bipartitionnement

Soient X et Y les deux ensembles sur lesquels est défini le tableau de données (Tab.2.1). On considère S l'ensemble de tous les couples de partitions de X et de Y. Une fonction $W : S \rightarrow R^+$, appelée *Critère*, mesurant la qualité d'un couple de partitions et dépendant du type de tableau de données est définie. Le problème de bipartitionnement consiste à déterminer l'élément S optimisant ce critère, c'est à dire le couple de partitions de X et Y qui minimisera ou maximisera suivant le cas la fonction W.

Soit A une matrice des données à N lignes et M colonnes, définie par l'ensemble $\{X = x_1, \ldots, x_N\}$ des lignes et l'ensemble $\{Y = y_1, \ldots, y_M\}$ des colonnes.
$a_{ij}, 1 \leq i \leq N$ et $1 \leq j \leq M$, sont les éléments de la matrice A.

	y_1	$\ldots$	y_i	$\ldots$	y_M
x_1	a_{11}	$\ldots$	a_{1j}	$\ldots$	a_{1M}
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
x_i	a_{i1}	$\ldots$	a_{ij}	$\ldots$	a_{iM}
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
x_N	a_{N1}	$\ldots$	a_{Nj}	$\ldots$	a_{NM}

TAB. 1 – *Matrice des données*

Une classe ligne $A_{IY} = (I, Y)$ est un sous-ensemble $I = \{i_1, ..., i_k\}$ de lignes ($I \subseteq X$ et $k \leq N$) ayant un comportement similaire sous l'ensemble Y des colonnes. Une classe ligne (I, Y) peut être définie comme une sous-matrice $k \times M$. De même, une classe colonne $A_{XJ} = (X, J)$ est un sous-ensemble de colonnes $J = \{j_1, ..., j_s\}$ ($J \subseteq Y$ et $s \leq M$) ayant un comportement similaire sous l'ensemble X des lignes. Cette classe colonne (X, J) peut être définie comme une sous-matrice $N \times s$. Une biclasse $A_{IJ} = (I, J)$ tels que $I = \{i_1, ..., i_k\}$ ($I \subseteq X$ et $k \leq N$) et $J = \{j_1, ..., j_s\}$ ($J \subseteq Y$ et $s \leq M$) est un sous ensemble de lignes ayant un comportement similaire sous un sous-ensemble de colonnes et vice-versa. Elle peut être définie comme une sous matrice $k \times s$. Les algorithmes de bipartitionnement ou de

classification simultanée ont pour objectif d'identifier un ensemble de biclasses $B_k = (I_k, J_k)$, I_k est une classe définie sur X et J_k est une classe définie sur Y, tel que chaque biclasse B_k satisfait certains critères d'homogénéité. Ces critères varient d'un algorithme à un autre.

2.2 Partitionnement et bipartitionnement

Contrairement aux méthodes de partitionnement simple qui s'appliquent séparément sur les lignes et les colonnes, les méthodes de bipartitionnement s'appliquent simultanément sur les deux dimensions et produisent des blocs homogènes dans lesquels chaque individu est caractérisé par un sous-ensemble d'attributs et chaque attribut caractérise un sous-ensemble d'individus (tab. 2.2).

Partitionnement	**Bipartitionnement**
- Appliqué aux lignes et aux colonnes **séparément**	- Appliqué aux lignes et aux colonnes **simultanément**
- Produit des classes (sous-groupes) sur les lignes **ou** des classes sur les colonnes	- Produit des biclasses (classes sur les lignes **et** sur les colonnes) i.e. des blocs homogènes
- Chaque objet est défini en utilisant **tous** les attributs et chaque attribut caractérise **tous** les objets. ⇒ On parle d'un Modèle **global**	- Chaque objet est défini en utilisant un **sous-ensemble** des attributs, chaque attribut caractérise un **sous-ensemble** d'objets. ⇒ On parle d'un Modèle **local**
- Les classes sont **exclusives**	- Les biclasses **ne sont pas exclusives/ exhaustives**

TAB. 2 – *Comparaison entre partitionnement et bipartitionnement*

Les biclasses dans ce modèle local ne sont ni exclusives ni exhaustives. En d'autres termes, certains individus et/ou variables n'appartiennent à aucune biclasse ou appartiennent à plusieurs biclasses à la fois alors que dans le partitionnement simple chaque individu est affecté à une et une seule classe. L'avantage du bipartitionnement réside principalement dans la découverte des corrélations importantes entre des sous-ensembles de lignes et de colonnes. Parmi les premiers travaux témoignant de l'intérêt des méthodes de bipartitionnement, ceux de Hartigan (1975), Govaert (1983), Bock (1979), et Marchotorchino (1987). Plus récemment plusieurs auteurs se sont intéressés à ce type d'approches, citons par exemple les travaux de Vichi (2000), Vichi et Kiers (2001), Bock (2003), Dhillon (2001), Dhillon et al. (2003), Mechelen et al. (2004), Madeira et Oliveira (2004), Govaert et Nadif (2005) et Govaert et Nadif (2009).

2.3 Type et structure des biclasses

2.3.1 Type des biclasses

Les algorithmes de bipartitionnement permettent de découvrir quatre types de biclasses (Madeira et Oliveira, 2004).

- Biclasses à valeurs constantes : une biclasse à valeurs constantes est une sous-matrice (I, J) tel que $\forall i \in I, \forall j \in J, a_{ij} = \mu$. Cependant, dans le cas réel, compte tenu du bruit dans les données, la valeur a_{ij} se présente généralement sous la forme $\eta_{ij} + \mu$, où η_{ij} est le bruit associé à la valeur réelle de a_{ij}.
- Biclasses à valeurs constantes sur les lignes ou les colonnes : une biclasse à valeurs constantes sur les lignes est une sous-matrice (I, J) où toutes les valeurs a_{ij} de la biclasse sont obtenues en utilisant le modèle additif ou le modèle multiplicatif suivant.
  - Modèle additif : $a_{ij} = \mu + \alpha_i$
  - Modèle multiplicatif : $a_{ij} = \mu \times \alpha_i$

  μ est une valeur caractéristique de la biclasse et α_i est l'ajustement sur la ligne $i \in I$. Une biclasse à valeurs constantes sur les colonnes est caractérisée par un ajustement β_j sur la colonne $j \in J$.
- Biclasses à valeurs cohérentes : une biclasse à valeurs cohérentes est définie en utilisant le modèle additif ou le modèle multiplicatif.
  - Modèle additif : $a_{ij} = \mu + \alpha_i + \beta_j$
  - Modèle multiplicatif : $a_{ij} = \mu \times \alpha_i \times \beta_j$

  où μ est une valeur caractéristique de la biclasse, α_i est l'ajustement sur la ligne $i \in I$ et β_j est l'ajustement sur la colonne $j \in J$.
- Biclasses à évolutions cohérentes : les approches visant à identifier dans les données des biclasses à évolutions cohérentes considèrent que les éléments de la matrice sont des valeurs symboliques et essayent de découvrir des sous-ensembles de lignes et des sous-ensembles de colonnes dont l'évolution est cohérente par rapport à la valeur exacte de la matrice des données.

2.3.2 Structure des biclasses

Certains algorithmes de bi-partitionnement cherchent dans les données une seule biclasse, d'autres ont pour objectif d'identifier plusieurs biclasses dont le nombre est généralement fixé *a priori*. Les biclasses découvertes dans une matrice des données peuvent avoir l'une des structures suivantes :

- Biclasses exclusives sur les lignes et les colonnes : les algorithmes produisant cette structure, DCC (Double Conjugated Clustering) (Busygin et al., 2002) par exemple, supposent que chaque ligne et chaque colonne de la matrice des données appartient exclusivement à une seule biclasse (Fig. 1(b)).
- Structure d'échiquier sans recouvrement : certains algorithmes tels que l'algorithme spectral Kluger et al. (2003) considèrent que les lignes et les colonnes peuvent appartenir à plus d'une biclasse tout en présentant la structure d'un échiquier. Ils construisent ainsi K biclasses non exclusives mais sans recouvrement. Chaque ligne appartient exactement à K biclasses (Fig. 1(c)).

– Biclasses exclusives sur les lignes : dans ce genre de biclasses, chaque ligne ne peut appartenir qu'à une seule biclasse. Par contre, chaque colonne peut appartenir à une ou plusieurs biclasses (Fig. 1(d)).
– Biclasses exclusives sur les colonnes : contrairement à la structure précédente, chaque colonne ne peut appartenir qu'à une seule biclasse alors que les lignes peuvent appartenir à une ou plusieurs biclasses (Fig. 1(e)).
– Biclasses sans recouvrement présentant une structure arborescente (Fig. 1(f)).
– Biclasses non exclusives sans recouvrement : dans les structures précédentes, les biclasses sont exhaustives. En effet, chaque ligne et chaque colonne de la matrice des données appartient à au moins une biclasse. Dans les biclasses non exclusives sans recouvrement, chaque pair (ligne, colonne) appartient à une et une seule biclasse (Fig. 1(g)).
– Biclasses avec recouvrement et présentant une structure hiérarchique : dans cette structure, les biclasses sont soit disjointes soit contenues l'une dans l'autre (Fig. 1(h)).
– Biclasses positionnées arbitrairement avec recouvrement : cette structure est la plus générale. Elle permet le recouvrement des biclasses et qu'une biclasse soit contenue dans une autre. En plus, dans cette structure, les biclasses ne sont pas exhaustives puisque certaines lignes ou colonnes n'appartiennent à aucune biclasse (Fig. 1(i)).

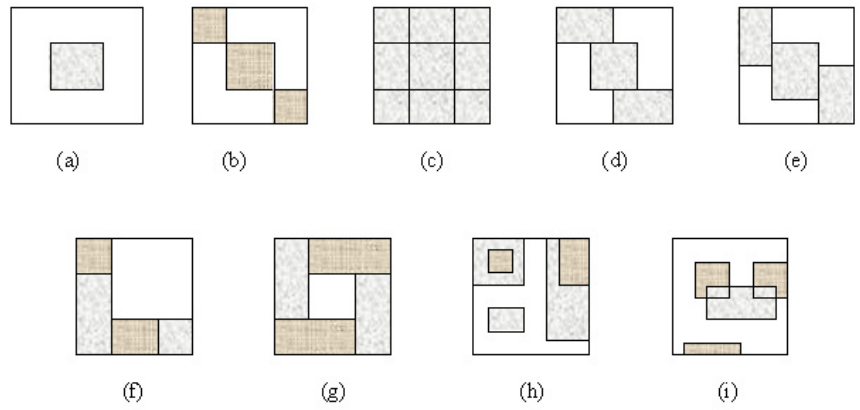

FIG. 1 – *Structure des biclasses.*

2.4 Approches de bipartitionnement

Dans Madeira et Oliveira (2004), les auteurs proposent de classifier les approches utilisées pour la construction des biclasses en cinq catégories.

– IRCCC (Iterative Row and Column Clustering Combination) : le principe de cette approche est d'appliquer un algorithme de partitionnement simple (kmeans, CAH,...) sur les lignes et les colonnes de la matrice séparément puis de combiner les résultats pour construire des biclasses.

- DC (Divide and Conquer) : cette approche affecte initialement tous les éléments à la même biclasse et procède à un découpage itératif. Elle présente l'avantage d'être rapide à identifier les biclasses dans les données. Cependant, elle a comme inconvénient de ne pas pouvoir garder certaines bonnes biclasses avant leur découpage.
- GIS (Greedy Iterative Search) : l'approche GIS est basée sur l'idée de créer des biclasses en ajoutant ou supprimant des lignes ou des colonnes afin d'optimiser un certain critère. Malgré sa rapidité, cette approche présente l'inconvénient de ne pas garder certaines bonnes biclasses en ajoutant ou supprimant des lignes ou des colonnes.
- EBE (Exhaustive Bicluster Enumeration) : l'approche EBE est basée sur l'idée que l'identification des meilleures biclasses nécessite une énumération exhaustive de toutes les biclasses possibles dans la matrice des données. L'avantage de cette approche est la certitude de trouver les meilleures biclasses mais son inconvénient majeur est sa complexité qu'il faut réduire en appliquant des restrictions sur la taille des biclasses.
- DPI (Distribution Parameter Identification) : l'approche DPI suppose la présence d'un modèle statistique dans les données et cherche à identifier la distribution des paramètres utilisés pour générer les données en minimisant certains critères à travers une approche itérative.

Besson (2005) propose de classifier les approches de bipartitionnement en quatre grandes classes de méthodes heuristiques : les méthodes agglomératives, les méthodes divisives, les méthodes par permutation et les méthodes par approximation de paramètres. La première classe de méthodes commence avec une partition discrète des lignes et/ou des colonnes puis les éléments de cette partition sont affectés successivement aux différents groupes. Ces groupes forment finalement les bipartitions. Les méthodes divisives débutent avec un motif formé de toutes les lignes et toutes les colonnes. Ce motif est par la suite découpé successivement en bi-ensembles plus petits. Les méthodes par permutation déplacent des éléments entre groupes afin d'améliorer la qualité globale de la classification. Cette amélioration est bien souvent locale de sorte que la collection finale n'est souvent qu'un optimum local. La dernière classe de méthodes est assez différente. Elle cherche à calculer des paramètres tels que la moyenne et l'écart type, et s'applique souvent aux modèles probabilistes ou statistiques.

2.5 Algorithmes de bipartitionnement

Plusieurs algorithmes de bipartitionnement ont été proposés selon le type des données. La majorité des algorithmes proposés pour le bipartitionnement des tableaux de mesure, tels que CTWC (Coupled two-way clustering) (Getz et al., 2000), ITWC "Interrelated Two-Way Clustering" (Zhang et al., 2001), DCC "Double Conjugated Clustering" (Busygin et al., 2002), δ-biclusters (Cheng et Church, 2000), OPSM "Order-Preserving Sub-Matrix" (Ben-Dor et al., 2002), δ-patterns (Califano et al., 2000), algorithme de Lazzeroni et Owen connu sous le nom de "Plaid models", l'algorithme spectral (Kluger et al., 2003), sont développés pour l'analyse des biopuces en bioinformatique. D'autres algorithmes, tels que CROEUC (Govaert, 1983), Block-EM et Block-CEM (Govaert et Nadif (2003), Govaert et Nadif (2005)) sont appliqués sur des tableaux de mesure issus de domaines différents. Les deux algorithmes Block-EM et Block-CEM, qui constituent une extension de l'algorithme EM, s'appliquent également à des tableaux de données binaires. Un autre algorithme, CROBIN (Classification **CRO**isée d'un tableau **BIN**aire), a été proposé par Govaert (1983) pour le bipartitionnement des données binaires. Quand aux tableaux de contingence, les algorithmes de bipartitionnement les

plus connus sont CROKI2 (classification **CRO**isée optimisant le **Khi2** du tableau de contingence) proposé par Govaert (1983), l'algorithme BSGP (Bipartite Spectral Graph Partitioning) proposé par Dhillon (2001), l'algorithme IT (Information-Theoretic co-clustering) de Dhillon et al. (2003) et l'algorithme Cemcroki2 proposé par Nadif et Govaert (2005).

Algorithme	**Fixer le NB. de Biclasses**	**Type de biclasses**	**Tableau de données**
One-way splitting	Non	Val. constantes	Tab. de mesures
Two-way splitting	Oui	Val. constantes	Tab. de mesures
CROEUC	Oui	Val. cohérentes	Tab. de mesures
CROKI2	Oui	Val. cohérentes	Tab. de contingence
CROBIN	Oui	Val. cohérentes	Tab. de données binaires
CTWC	Non	Val. constantes sur les colonnes	Tab. de mesures
Plaid Models	Oui	Val. cohérentes	Tab. de mesures
δ-biclusters	Oui	Val. cohérentes	Tab. de mesures
δ-patterns	Non	Val. constantes sur les lignes	Tab. de mesures
ITWC	Non	Val. cohérentes	Tab. de mesures
DCC	Non	Val. constantes	Tab. de mesures
OPSM	Non	Evolution cohérente	Tab. de mesures
SAMBA	Non	Evolution cohérente	Tab. de mesures
FLOC	Oui	Val. cohérentes	Tab. de mesures
Spectral	Non	Val. cohérentes	Tab. de mesures
IT	Oui	Val. cohérentes	Tab. de contingence
BSGP	Oui	Val. cohérentes	Tab. de contingence
Robardet (2002)	Non	Val. cohérentes	Tab. de contingence
Block-EM	Oui	Val. cohérentes	Tab. de mesures Tab. de données binaires
Block-CEM	Oui	Val. cohérentes	Tab. de mesures Tab. de données binaires
Cemcroki2	Oui	Val. cohérentes	Tab. de contingence

TAB. 3 – *Caractéristiques des algorithmes de bipartitionnement.*

Les algorithmes de bipartitionnement diffèrent selon l'approche sur laquelle ils reposent, le type de biclasses obtenues, le type de tableau des données et le domaine d'application (tab. 2.5). La majorité de ces algorithmes, à savoir CTWC, δ-biclusters, δ-patterns, ITWC, DCC, OPSM, SAMBA, FLOC et l'algorithme spectral, ont été développés pour l'analyse des biopuces en bioinformatique, en particulier pour identifier des groupes de gènes ayant le même comportement sous un groupe de conditions expérimentales. Les autres algorithmes, ont été développés pour des applications différentes. Les deux algorithmes proposés par Dhillon dans Dhillon (2001) et Dhillon et al. (2003) ont été appliqués à des tableaux de contingence croisant des documents à des mots-clés pour la catégorisation des documents.

Certains de ces algorithmes présentent l'avantage de pouvoir identifier les biclasses sans avoir besoin de fixer *a priori* le nombre de biclasses ou le nombre de classes sur les lignes et les colonnes. D'autres, par contre, nécessitent d'avoir une idée sur la structure des données.
Dans ce papier, nous nous intéressons aux algorithmes destinés au bipartitionnement des tableaux de contingence.

2.6 Algorithmes de bipartitionnement des tableaux de contingence

Algorithme CROKI2

L'algorithme CROKI2 (classification **CRO**isée optimisant le **Khi2** du tableau de contingence) proposé par Govaert (1983) a pour objectif de trouver une partition P de X en K classes et une partition Q de Y en L classes telle que le Khi2 de contingence du nouveau tableau construit en regroupant les lignes et les colonnes suivant les partitions P et Q soit maximum. L'algorithme CROKI2 consiste à déterminer une série de couples de partitions (P_n, Q_n) optimisant le Khi2 du tableau de contingence en appliquant alternativement sur X et sur Y une variante de la méthode des nuées dynamiques.

Algorithme BSGP

L'algorithme BSGP (Bipartite Spectral Graph Partitioning) proposé par Dhillon (2001) utilise la théorie spectrale des graphes pour la représentation des documents et des mots-clés. Les sommets du graphe sont les documents et les mots-clés et les arêtes sont les liaisons entre les mots-clés et les documents dans lesquels ils apparaissent. Le poids de chaque arête correspond au nombre d'occurrences du mot-clé dans le document. Le problème de bipartitionnement correspond alors au problème de partitionnement spectral d'un graphe biparti. L'algorithme BSGP fait appel à la décomposition en valeurs singulières et à l'algorithme de Kmeans pour l'identification des classes des documents et des mots-clés. Cet algorithme, comme tous les algorithmes basés sur le Kmeans, nécessite de fixer le nombre de classes dans les données.

Algorithmes basées sur la théorie de l'information

Deux variantes d'une même méthode de bipartitionnement ont été développées d'une manière indépendante par Dhillon et al. (2003) et Robardet (2002). Cette méthode consiste à considérer les deux partitions recherchées comme deux variables aléatoires à valeurs discrètes et à concevoir la recherche d'une bipartition comme un problème de maximisation de l'association entre ces deux variables. Les deux variantes produisent une partition par un processus d'optimisation locale. L'algorithme IT (Information-Theoretic co-clustering) Dhillon et al. (2003) utilise la mesure de divergence entre la distribution de probabilités de Kullback et Leibler et propose de fixer a priori le nombre de classes de chacune des deux partitions puis optimise localement la fonction en estimant itérativement une partition en fonction de l'autre jusqu'à la convergence. L'algorithme de Robardet et al. ne nécessite pas de fixer a priori le nombre de classes des deux partitions. Il utilise un algorithme d'optimisation locale stochastique qui procède également par ajustement itératif d'une partition en fonction de l'autre.

Algorithme Cemcroki2

L'algorithme Cemcroki2 proposé par Nadif et Govaert (2005) est une extension de l'algorithme Croki2 basée sur un modèle de mélange croisé de distributions de Poisson. L'algo-

rithme Cemcroki2 repose sur l'utilisation de l'algorithme CEM "Classification EM" pour la classification alternée des lignes et des colonnes au lieu de l'algorithme des nuées dynamiques utilisé dans Croki2. Son avantage par rapport à Croki2 est qu'il s'applique mieux aux données réelles où les classes ne sont pas bien séparées et les proportions des classes sont différentes. Des expérimentations effectuées par les auteurs sur des données réelles ont montré aussi qu'il fournit des meilleurs résultats par rapport algorithmes IT et BSGP proposés par Dhillon avec un nombre de documents mal classés nettement inférieur. Cependant, comme tous ces algorithmes, l'algorithme Cemcroki2 nécessite de fixer le nombre de classes sur les lignes et les colonnes.

3 Algorithme Croki2 de bipartitionnement d'un tableau de contingence

Soit A un tableau de contingence à N lignes et M colonnes, défini par l'ensemble $\{X = x_1, \ldots, x_N\}$ des lignes et l'ensemble $\{Y = y_1, \ldots, y_M\}$ des colonnes. a_{ij}, $1 \leq i \leq N$ et $1 \leq j \leq M$, sont les éléments du tableau A.

Le χ^2 du tableau de contingence initial s'écrit :

$$\chi^2(X,Y) = S \sum_{i \in X} \sum_{j \in Y} \frac{(f_{ij} - f_{i.} f_{.j})^2}{f_{i.} f_{.j}}$$

avec

$$S = \sum_{i \in X} \sum_{j \in Y} a_{ij}$$

$$f_{ij} = \frac{a_{ij}}{S}, f_{i.} = \sum_{j \in Y} f_{ij}, f_{.j} = \sum_{i \in X} f_{ij}$$

L'algorithme Croki2 consiste à trouver une partition $P = (P_1, ..., P_K)$ de X en K classes et une partition $Q = (Q_1, ..., Q_L)$ de Y en L classes telles que le χ^2 de contingence du nouveau tableau de données $T(P, Q)$ soit maximum. À partir d'un couple initial (P^0, Q^0), une suite de couples de partitions (P^n,Q^n) optimisant le χ^2 de contingence est construite.

Le critère général à optimiser est :

$$\chi^2(P,Q) = \sum_{k=1}^{K} \sum_{l=1}^{L} \frac{(f_{kl} - f_{k.} f_{.l})^2}{f_{k.} f_{.l}}$$

avec

$$f_{kl} = \sum_{i \in P_k} \sum_{j \in Q_l} f_{ij}$$

$$f_{k.} = \sum_{l=1,L} f_{kl} = \sum_{i \in P_k} f_{i.}$$

$$f_{.l} = \sum_{k=1,K} f_{kl} = \sum_{j \in Q_l} f_{.j}$$

$$S = \sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij}$$

	Q_1	...	Q_L
P_1	...	...	...
...	...	g_{kl}	...
P_K	...	...	...

TAB. 4 – *Nouveau tableau de contingence construit en regroupant les lignes et les colonnes suivant les partitions P et Q.*

L'élément général du tableau $T(P,Q)$ est $g_{kl} = \sum_{i \in P_k} \sum_{j \in Q_l} a_{ij}$.

3.1 Déroulement de l'algorithme Croki2

L'algorithme Croki2 se déroule comme suit :

Pour chaque tirage aléatoire des partitions initiales (P^0, Q^0)
1 Démarrer de la position initiale $(P^{(0)}, Q^{(0)}, G^{(0)})$
2 Calculer $(P^{(n+1)}, Q^{(n+1)}, G^{(n+1)})$ à partir $(P^{(n)}, Q^{(n)}, G^{(n)})$
3 Calculer $(P^{(n+1)}, Q^{(n)}, G^{(n+\frac{1}{2})})$ à partir de $(P^{(n)}, Q^{(n)}, G^{(n)})$
4 Calculer $(P^{(n+1)}, Q^{(n+1)}, G^{(n+1)})$ à partir de $(P^{(n+1)}, Q^{(n)}, G^{(n+\frac{1}{2})})$
5 Recommencer l'étape 2 jusqu'à la convergence de l'algorithme

La recherche des deux partitions (P,Q) sur les lignes et les colonnes peut se faire selon plusieurs stratégies. La stratégie proposée par Govaert (1983) consiste à alterner l'optimisation sur les lignes puis sur les colonnes en utilisant l'algorithme des nuées dynamiques jusqu'à la convergence.

– **Optimisation sur les lignes**
On bloque la partition en colonnes Q et on ne travaille que sur la partition P en lignes. Soit le tableau de contingence $A(X,Q)$ défini par :

$$A_1(x,l) = \sum_{j \in Q_l} a_{ij}$$

où $Q = (Q_1, ..., Q_L)$. En considérant que les objets à classer sont les éléments de X et les variables les classes de Q, une suite de partitions de X en L classes est construite

telle que les χ^2 de contingence associés soient croissants jusqu'à la convergence. Le critère à optimiser avec l'algorithme des nuées dynamiques simple est :

$$\Delta(P/Q,G) = \sum_{k=1}^{K} \sum_{i \in P_k} d_{\chi^2}(u_i, G_k)$$

– **Optimisation sur les colonnes**
Contrairement à l'étape précédente, on fixe la partition en lignes P et on applique l'algorithme des nuées dynamiques sur la partition en colonnes Y. Le tableau de contingence est cette fois $A(P,Y)$ défini par :

$$A_1(k,y) = \sum_{i \in P_k} a_{ij}$$

où $P = (P_1, ..., P_K)$. Une suite de partitions de Y en L classes fait croître le χ^2 jusqu'à la convergence. Cette fois, on cherche à optimiser le critère suivant :

$$\Delta(Q/P,G) = \sum_{l=1}^{L} \sum_{j \in Q_l} d_{\chi^2}(v_j, G_l)$$

3.1.1 Problème d'optimisation

L'algorithme Croki2 recherche alternativement une partition de X et une partition de Y en appliquant à chaque fois l'algorithme des nuées dynamiques sur une partition en fixant l'autre et vice versa. Sur le tableau de contingence $A(X,Q)$, plusieurs itérations sont nécessaires pour atteindre la convergence de l'algorithme des nuées dynamiques. Une suite de partitions de X, $H^0, ..., H^r$, en K classes est construite telle que les χ^2 de contingence associés soient croissants jusqu'à la convergence.

$$\chi^2(H^0,Q^n) < \chi^2(H^1,Q^n) < ... < \chi^2(H^r,Q^n) = \chi^2(H^{r+1},Q^n)$$

$$\chi^2(H^r,Q^n) = \Delta(H^r/Q^n,G) = \Delta(Q^n/H^r,G)$$

où

$$G(k,l) = \sum_{j \in Q_l^n} \sum_{i \in H_k^r} a_{ij}$$

r est l'étape à partir de laquelle la suite devient stationnaire. La partition P^{n+1} sera la partition obtenue à la convergence : $P^{n+1} = H^r$. De même pour le tableau de contingence $A(P,Y)$, plusieurs itérations sont nécessaires pour atteindre la convergence de l'algorithme des nuées dynamiques. À un niveau supérieur, une autre boucle d'itérations sert à alterner l'optimisation sur les lignes et l'optimisation sur les colonnes jusqu'à ce que l'ensemble des deux partitions soit stable. Ainsi, pour chaque tirage, l'algorithme Croki2 est itératif à deux niveaux.

3.1.2 Algorithme Croki2

```
Pour chaque tirage aléatoire des partitions initiales
    Démarrer d'une position initiale (P^0, Q^0, G^0)
        Pour iter < iterMax
            Optimisation sur les lignes
                Etape de représentation
                Etape d'affectation par les nuées dynamiques
            Optimisation sur les colonnes
                Etape de représentation
                Etape d'affectation par les nuées dynamiques
        FinPour
FinPour
```

4 Accélération de l'algorithme Croki2

La stratégie utilisée dans l'algorithme Croki2 proposé par Govaert (1983) est l'optimisation alternée de la partition en lignes et la partition en colonnes. Dans ce papier, nous proposons une nouvelle variante de l'algorithme Croki2 qui consiste à combiner les deux étapes d'optimisation utilisées dans la version originale de Croki2 en une seule étape. L'algorithme des nuées dynamiques est remplacé par une simple affectation de l'individu à la classe la plus proche. Une boucle d'itérations permet d'alterner les étapes d'affectation et de représentation sur les lignes et les colonnes plusieurs fois jusqu'à la convergence. La convergence est atteinte lorsqu'aucun individu sur les lignes et sur les colonnes ne change de classe d'appartenance.

4.1 Algorithme Croki2 accéléré

L'algorithme suivant explicite le déroulement de l'algorithme Croki2 accéléré.

```
Pour chaque tirage aléatoire des partitions initiales
    Démarrer d'une position initiale (P^0, Q^0, G^0)
            Etape d'affectation des lignes
            Etape de représentation des lignes
            Etape d'affectation des colonnes
            Etape de représentation des colonnes
FinPour
```

Les étapes de représentation et d'affectation sont définies comme suit :

– Étape d'affectation des lignes : consiste à calculer pour chaque objet i de X l'indice k^* de la classe d'affectation qui vérifie

$$k^* = argmin_{k=1\ldots K} d_{\chi^2}(u_i, G_k)$$

– Étape de représentation des lignes : consiste à calculer pour chaque classe k le prototype $G_k = (g_{k1}, ..., g_{kl}, ..., g_{kL})$

$$g_{kl} = \sum_{i \in P_k} a_{il} = \sum_{i \in P_k} \sum_{j \in Q_l} a_{ij}$$

– Étape d'affectation des colonnes : consiste à calculer pour chaque objet j de Y l'indice l^* de la classe d'affectation qui vérifie

$$l^* = argmin_{l=1...L} d_{\chi^2}(v_j, G_l)$$

– Étape de représentation des colonnes : consiste à calculer pour chaque classe l le prototype $G_l = (g_{1l}, ..., g_{kl}, ...g_{Kl})$

$$g_{kl} = \sum_{j \in Q_l} a_{kj} = \sum_{j \in Q_l} \sum_{i \in P_k} a_{ij}$$

Sachant que l'algorithme Croki2 débute avec deux partitions initiales tirées au hasard sur les lignes et les colonnes $^(P^0, Q^0)$, les résultats obtenus, comme pour toutes les méthodes convergeant vers un optimum local, dépendent de ces partitions initiales. Il est donc nécessaire d'exécuter plusieurs fois l'algorithme afin d'assurer l'indépendance du résultat final des partitions initiales.

4.2 Calcul de la complexité des deux algorithmes

L'algorithme Croki2 dans sa version originale est itératif à deux niveaux. Le premier niveau d'itérations assure la convergence de l'algorithme des nuées dynamiques. Soient, pour un couple de classes (K, L) donné, $niter11$ et $niter12$ le nombre d'itérations nécessaires pour la convergence de l'algorithme des nuées dynamiques sur les lignes et sur les colonnes respectivement, $niter2$ le nombre d'itérations nécessaires pour alterner l'optimisation sur les lignes et l'optimisation sur les colonnes. Comme l'opération la plus coûteuse dans l'algorithme Croki2 est le calcul des distances, nous estimons le nombre total de calcul de distances dans l'algorithme. Soit le tableau des données de dimensions (N, M). Pour un tirage donné, et pour chaque itération au niveau supérieur (allant de 1 à niter2), il est nécessaire d'effectuer $niter11 \times N \times K$ calculs de distances dans R^L et $niter12 \times M \times L$ calculs de distances dans R^K. Or niter11 et niter12 varient d'une itération à une autre dans la boucle supérieure. Ainsi, le nombre total d'opérations pour Croki2 dans sa version originale est donné par la formule suivante pour un tirage donné :

$$Croki2 : \sum_{i=1}^{niter2} \times (niter11_i \times N \times K \times L + niter12_i \times M \times L \times K)$$

$$Croki2 : K \times L \times \sum_{i=1}^{niter2} (niter11_i \times N + niter12_i \times M)$$

L'algorithme Croki2 accéléré présente un seul niveau d'itérations. Soit $niter$ le nombre d'itérations nécessaires pour la convergence de l'algorithme pour un couple de classes (K, L) donné. Le nombre total d'opérations dans ce cas est :

$$Croki2accelere : niter \times (N \times K \times L + M \times L \times K) = niter \times K \times L \times (N + M)$$

Il est clair d'après ces deux formules que l'algorithme Croki2 accéléré est plus rapide que l'algorithme Croki2.

4.3 Comparaison des deux algorithmes

Nous comparons la nouvelle variante de l'algorithme Croki2 à la version originale de l'algorithme à l'aide d'un ensemble de critères tels que la complexité, l'aptitude à déterminer la solution optimale et leur performance dans la génération de partitions avec des classes non vides. Pour ce faire, nous utilisons des jeux des données que nous générons artificiellement.

4.3.1 Génération des données artificielles

Soient (x_1,...,x_i,...,x_n) et (y_1,...,y_j,...,y_p) les ensembles artificiels de n individus et p variables que nous voulons créer sur les lignes et les colonnes respectivement.

Ainsi, à chaque ligne i un vecteur x_i de colonnes est associé et à chaque colonne un vecteur y_j de lignes est attribué, avec $x_i = (x_i^1, ..., x_i^j, ..., x_i^p)$ et $y_j = (y_j^1, ..., y_j^i, ..., y_j^n)$.

Soient (p_1,...,p_k,...,p_K) le vecteur des profils initiaux des K classes *a priori* sur les lignes et (q_1,...,q_l,...,q_L) le vecteur des profils initiaux des L classes *a priori* sur les colonnes. La répartition des individus sur les classes *a priori* est effectuée selon les pourcentages indiqués par le vecteur α sur les lignes et le vecteur β sur les colonnes, avec α = (α_1,...,α_k,...α_K) et β=(β_1,...,β_l,...β_L), $\alpha_k \in [0, 1]$, $\beta_l \in [0, 1]$ et

$\sum_{k=1}^{K} \alpha_k = 1$ et $\sum_{l=1}^{L} \beta_l = 1$

Algorithme de génération des données artificielles

L'algorithme que nous proposons commence par générer à partir du tableau des profils initiaux de dimensions (K,L) un nouveau tableau de profils sur les colonnes de dimensions (K, p=NombreDeVariables).

Pour chaque classe l sur les colonnes
  Pour chaque variable j de classe l
    q_j = q_l

L'étape suivante consiste à générer les données artificielles sur les lignes à partir du nouveau tableau des profils. Les entrées de l'algorithme permettant cette génération sont :

- NbOccMin = Nombre Minimum d'Occurrences
- NbOccMax = Nombre Maximum d'Occurrences
- NbIndivLig = Nombre d'Individus sur les Lignes
- NbIndivCol = Nombre d'Individus sur les Colonnes

Un entier compris entre $NbOccMin$ et $NbOccMax$ est tiré au hasard et attribué à la variable $NbOcc_i$. Cet entier représente les profils marginaux sur les lignes (c.à.d. le nombre total d'occurrences sur chaque ligne i). Ce nombre d'occurrences devrait être distribué sur les colonnes en respectant la distribution des occurrences imposée par les prototypes initiaux des classes. C'est la raison de l'introduction de la fonction $cumul$ sur la fréquence des occurrences dans les prototypes des classes.

```
Pour chaque classe k sur les lignes, k dans l'intervalle [1,K]
Pour chaque individu i de la classe ligne k
    NbOcc_i ← random(NbOccMin, NbOccMax)
    Pour j dans l'intervalle [1, NbOcc_i]
        y ← random(0,1)
        Pour l dans l'intervalle [1,L]
            Si y ≤ cumul (P_k^l)
                x_i^l = x_i^l + 1
        FinPour
    FinPour
FinPour
FinPour
```

Nous fixons la dimension des tableaux générés à 200 lignes et 100 colonnes. Le tableau 4.3.1 présente 6 jeux des données composés de classes bien séparées sur les lignes et les colonnes que nous utilisons dans nos expérimentations.

	Nombre de classes sur les lignes	Nombre de classes sur les colonnes
JD3x3	3	3
JD4x4	4	4
JD5x4	5	4
JD6x3	6	3
JD3x8	3	8
JD6x6	6	6

TAB. 5 – *Jeux des données avec biclasses séparées.*

Une projection des classes-lignes et des classes-colonnes des jeux des données JD3x3, JD5x4 et JD6x3 sur le premier plan factoriel d'une ACP est présentée sur la figure 2.

4.3.2 Comparaison du χ^2 à la convergence

Dans l'algorithme Croki2, un bon bipartitionnement est obtenu pour des valeurs élevées de χ^2 reflétant une corrélation maximale entre la partition des lignes et la partition des colonnes.

Notre idée est de comparer les valeurs de χ^2 obtenues à la convergence par les deux algorithmes Croki2 et Croki2 accéléré sachant que les deux algorithmes sont initialisés de la même manière et que la même stratégie de tirage aléatoire est appliquée dans les deux cas. En d'autres termes, nous imposons aux deux algorithmes de commencer avec la même partition

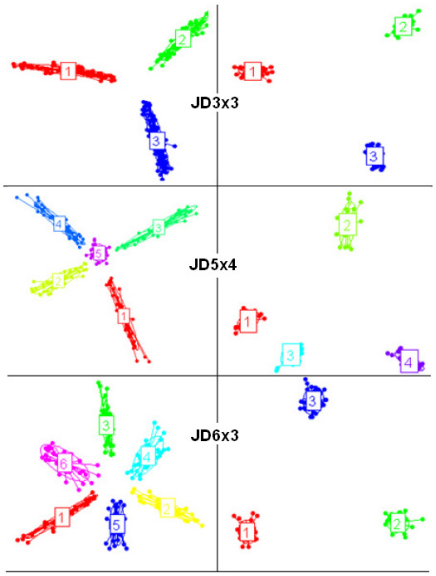

FIG. 2 – *Projection des biclasses des jeux de données JD3x3, JD5x4 et JD6x3 sur le premier plan factoriel d'une ACP. À gauche c'est la projection des classes sur les lignes et à droite c'est la projection des classes sur les colonnes.*

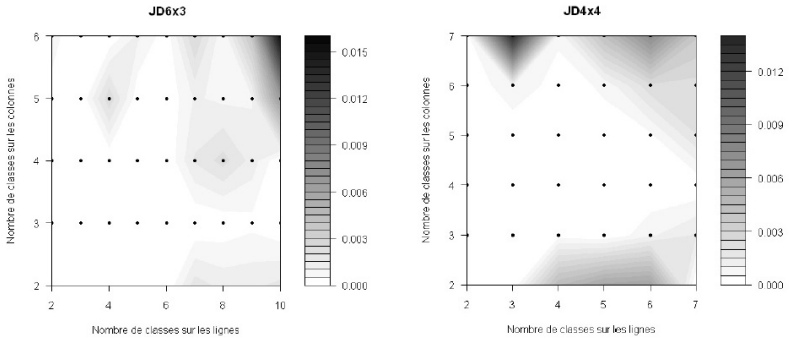

FIG. 3 – *Représentation de la différence entre les valeurs de χ^2 obtenues à la convergence par les deux algorithmes appliqués à JD6x3 et JD4x4.*

initiale sur les lignes et sur les colonnes (même conditions initiales).
Nous proposons alors de calculer la différence entre les valeurs de χ^2 obtenues par Croki2 et celles obtenues par Croki2 accéléré.

$$Diff = \frac{2 \times \left|\chi^2_{Croki2} - \chi^2_{Croki2Accelere}\right|}{(\chi^2_{Croki2} + \chi^2_{Croki2Accelere})}$$

La représentation graphique de la différence entre les valeurs de χ^2 (fig.3) montre que cette différence n'est pas significative. Cette différence est nulle lorsque le nombre de classes sur les lignes et les colonnes correspond au bon nombre de classes ((4,4) dans le cas de JD4x4 et (6,3) dans le cas de JD6x3).

4.3.3 Comparaison des bipartitions

Pour comparer les partitions obtenues par les deux algorithmes, nous appliquons pour chaque couple de classes, l'indice de Rand corrigé et la F-mesure sur les partitions obtenues par les deux algorithmes sur les lignes (resp. les colonnes).

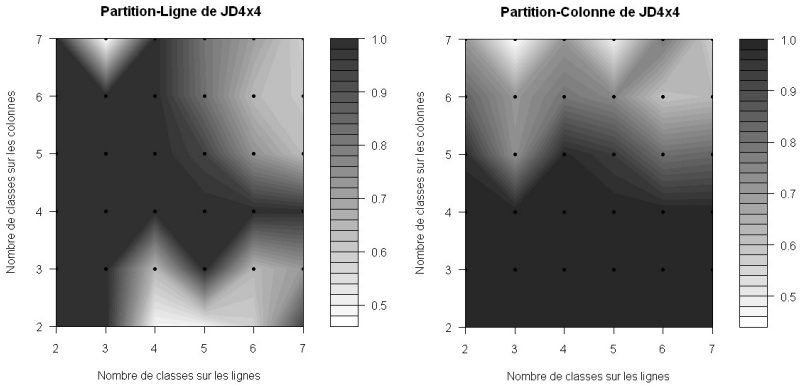

FIG. 4 – *Représentation de l'indice de Rand corrigé appliqué aux partitions-Lignes et partitions-Colonnes du jeu des données JD4x4 en fonction du nombre de classes sur les lignes et les colonnes.*

D'après les graphiques des figures 4 et 5, nous remarquons que plus on s'éloigne du bon nombre de classes, l'Indice de Rand corrigé s'éloigne de 1 donc la différence entre les partitions augmente. En effet, lorsque le nombre de classes recherchées sur les lignes (resp. les colonnes) augmente, les chances d'affecter deux individus à la même classe par les deux algorithmes diminuent. Par ailleurs, nous notons que, pour tous les jeux des données, les partitions retrouvées par les deux algorithmes coïncident lorsqu'on leur donne en entrée le bon nombre de classes sur les lignes et les colonnes. En d'autres termes, lorsque le nombre de classes en entrée est correct les deux algorithmes affectent les individus, aussi bien sur les lignes que sur les colonnes, aux mêmes classes (l'indice de Rand corrigé est égal à 1).

La F-mesure calculée sur les partitions-lignes et les partitions-colonnes obtenues par les deux algorithmes pour chaque couple de classe confirme les résultats obtenues par l'indice de Rand corrigé (fig.6 et fig.7). La représentation graphique de l'indice de Rand corrigé et la F-mesure appliqués aux partitions-lignes et partitions-colonnes des jeux des données JD6x3 et JD4x4 (fig.8) montre que les deux indices coïncident pour la majorité des couples de classes (k, l), en particulier lorsque ce dernier correspond au bon nombre de classes sur les lignes et les colonnes. Dans le cas où ils ont des valeurs différentes, la valeur de l'Indice de Rand corrigé est légèrement inférieure à celle de la F-mesure. Il s'avère alors que l'Indice de Rand corrigé

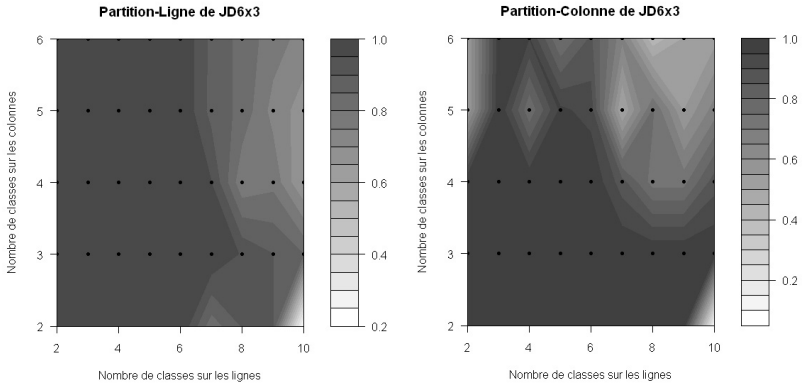

FIG. 5 – *Représentation de l'indice de Rand corrigé appliqué aux partitions-Lignes et partitions-Colonnes du jeu des données JD6x3 en fonction du nombre de classes sur les lignes et les colonnes.*

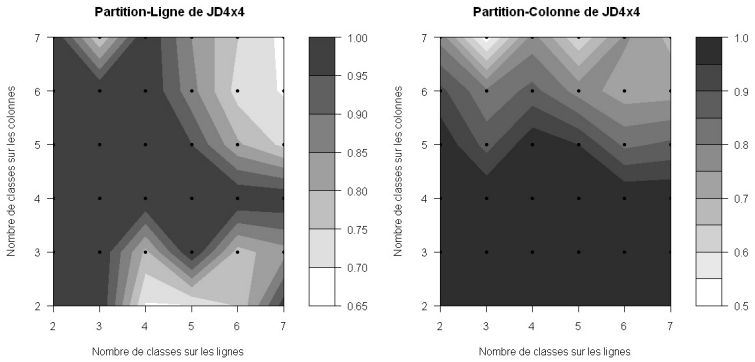

FIG. 6 – *Représentation de la F-mesure appliquée aux partitions-Lignes et partitions-Colonnes du jeu des données JD4x4 en fonction du nombre de classes sur les lignes et les colonnes.*

est plus sensible aux différences entre les deux partitions. Ceci s'explique aussi par le fait que l'Indice de Rand corrigé effectue une analyse globale des deux partitions comparées, alors que la F-mesure réalise une analyse plus fine classe par classe.

4.3.4 Performance dans l'identification de la structure initiale des données

Afin d'évaluer la performance de chacun des algorithmes dans l'identification de la structure initiale des données, nous proposons d'utiliser l'Indice de Rand corrigé. Pour ce faire, nous appliquons les deux algorithmes sur tous les jeux des données générées en faisant varier le nombre de classes sur les lignes et les colonnes. Pour chaque algorithme et pour chaque

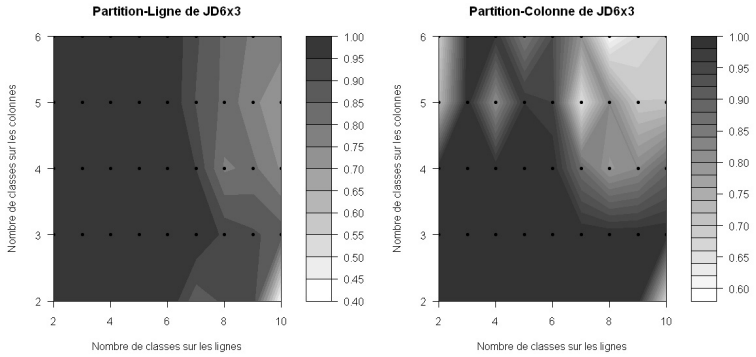

FIG. 7 – *Représentation de la F-mesure appliquée aux partitions-Lignes et partitions-Colonnes du jeu des données JD6x3 en fonction du nombre de classes sur les lignes et les colonnes.*

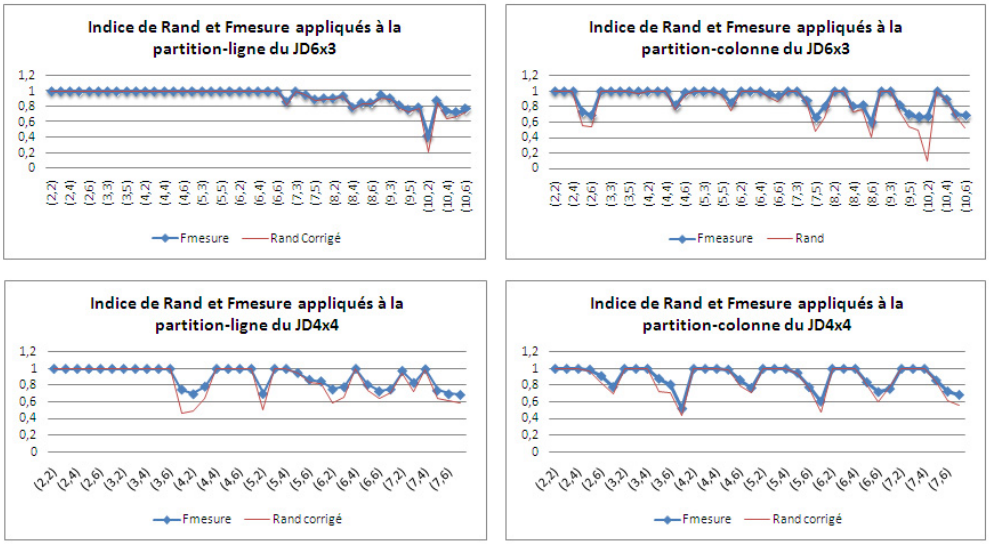

FIG. 8 – *Représentation de la F-mesure et l'Indice de Rand corrigé appliqués aux partitions-Lignes et partitions-Colonnes des jeux des données JD6x3 et JD4x4. Les valeurs de l'Indice de Rand corrigé sont égales ou légèrement inférieures à celles de la F-mesure.*

couple de classes nous récupérons la partition- ligne et la partition-colonne correspondante. Chaque partition est comparée à la partition initiale (c.à.d. artificielle) à travers l'indice de Rand corrigé. Une valeur maximale de l'indice (proche de 1) indique que les deux partitions sont égales. En d'autres termes, l'algorithme affecte les individus aux bonnes classes.
Soit l'exemple du jeu des données JD5x4. L'indice de Rand corrigé appliqué à la partition-

colonne prend la valeur 1 pour les couples $\{(3,4),(4,4),(5,4),(6,4),(7,4)\}$ (fig.9 et fig.10).

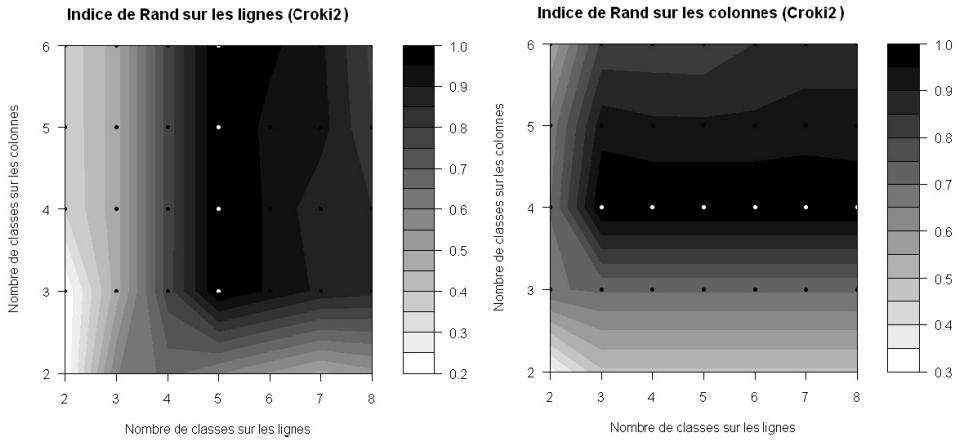

FIG. 9 – *Comparaison par l'indice de Rand corrigé des partitions réelles (i.e. générées artificiellement) aux partitions-lignes (resp. partitions-colonnes) résultant de l'application de Croki2 sur JD5x4.*

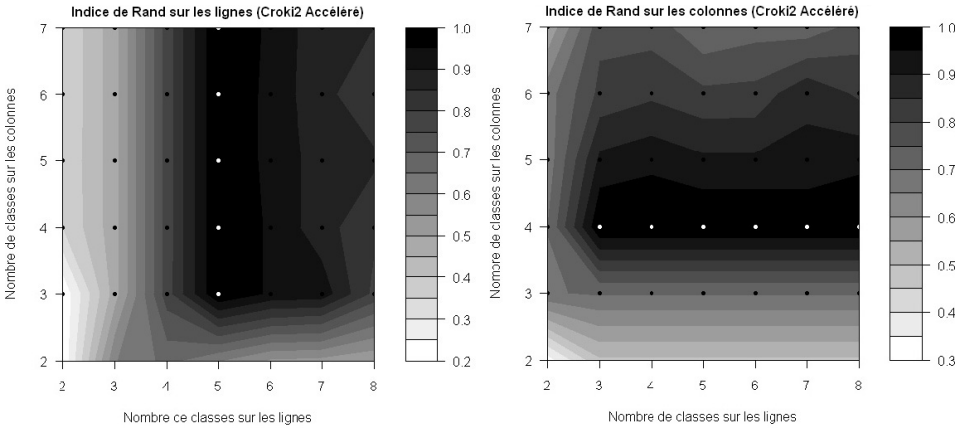

FIG. 10 – *Comparaison par l'indice de Rand corrigé des partitions réelles (i.e. générées artificiellement) aux partitions-lignes (resp. partitions-colonnes) résultant de l'application de Croki2 accéléré sur JD5x4.*

Ainsi, au niveau des colonnes, l'algorithme retrouve la bonne partition composée de 4 classes quelque soit le nombre de classes sur les lignes. De même, au niveau des lignes, l'indice de Rand corrigé prend la valeur 1 pour les couples $\{(5,3),(5,4),(5,5),(5,6),(5,7),(5,8)\}$. L'algorithme affecte les individus aux bonnes classes lorsque le nombre de classes sur les lignes est égal à 5 indépendamment du nombre de classes sur les colonnes. L'intersection des

deux ensembles de solutions est le couple de classes $(5, 4)$. Ceci montre bien que les deux algorithmes sont aptes à identifier le bon nombre de classes dans les données.

Le couple de classes qui maximise à la fois l'indice de Rand corrigé appliqué à la partition-ligne et celui appliqué à la partition-colonne est le $(5, 4)$ qui correspond au bon couple de classes. Pour tous les jeux des données, les deux indices de Rand corrigé obtenus sur la partition-ligne et la partition-colonne sont égaux à 1 lorsque le couple de classes coïncide avec le couple choisi dès le départ pour la construction des jeux des données, par exemple (3,3) pour JD3x3 et (5,4) pour JD5x4 (fig.11). Ainsi, lorsque le nombre de classes sur les lignes et les colonnes correspond au bon nombre de classes, les deux algorithmes, Croki2 et Croki2 accéléré, sont capables d'affecter les individus aux bonnes classes.

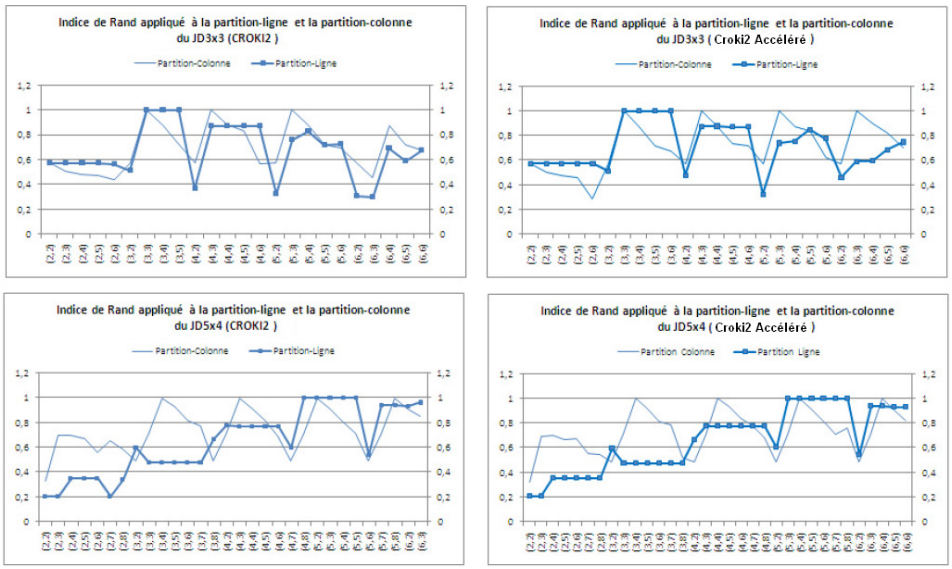

FIG. 11 – *Indice de Rand corrigé appliqué à la partition-ligne et la partition-colonne obtenues par les algorithmes Croki2 et Croki2 accéléré sur les jeux des données 5x4 et 3x3.*

4.3.5 Comparaison de la complexité des deux algorithmes

Pour comparer expérimentalement la complexité des deux algorithmes, nous les appliquons sur les jeux des données artificielles.

Dans l'exemple de la figure 12, nous fixons le nombre de classes sur les lignes à $K = 6$ et le nombre de classes sur les colonnes à $L = 3$ et nous examinons les quatre premiers tirages aléatoires des centres initiaux. Dans tous ces tirages, la complexité de l'algorithme Croki2 accéléré est nettement inférieure à celle de l'algorithme Croki2 dans sa version originale. Par exemple, dans le premier tirage, la complexité de Croki2 accéléré est égale à 32400 alors que celle de l'algorithme Croki2 original est égale à 68400. Dans le deuxième tirage, la complexité de Croki2 accéléré est égale à 54000 alors que celle de l'algorithme Croki2 original est égale

K	L	Tirage	niter	Khi2	complexité OG
6	3	1	6	13576.02	32400
6	3	2	10	12400.36	54000
6	3	3	6	13576.02	32400
6	3	4	4	12501.52	21600

K	L	Tirage	niter11	niter12	niter2	Khi2	complexité OA
6	3	1	14	3	1		
6	3	1	3	1	2	12387.86	68400
6	3	2	10	3	1		
6	3	2	4	1	2	12492.66	57600
6	3	3	10	2	1		
6	3	3	5	1	2	12528.51	59400
6	3	4	13	8	1		
6	3	4	5	2	2	11086.80	82800

FIG. 12 – *Exemple des données permettant le calcul de la complexité de Croki2 et Croki2 accéléré sur JD6x3.*

à 57600. Le calcul de la complexité des deux algorithmes sur les différents jeux des données artificielles montre que la complexité de l'algorithme Croki2 original dépasse celle de l'algorithme Croki2 accéléré. Par suite, l'algorithme Croki2 accéléré permet de gagner en temps de calcul, en particulier lorsque le nombre de classes fixé sur les lignes et les colonnes est grand.

5 Conclusion

Nous avons proposé dans ce papier une variante plus rapide de l'algorithme Croki2. Les expérimentations menées sur les données artificielles ont montré l'aptitude des deux algorithmes à identifier la structure initiale des données et à affecter les individus aux bonnes classes sur les lignes et les colonnes. D'autre part, la comparaison des résultats obtenus par les deux algorithmes montre que l'algorithme proposé est nettement plus rapide que l'algorithme Croki2 original. Comme perspective de ce travail, nous envisageons de tester les deux algorithmes sur des données réelles présentant une structure de biclasses et des données artificielles présentant un recouvrement de biclasses afin de mieux mesurer l'apport de l'accélération proposée.

Références

Ben-Dor, A., B. Chor, R. Karp, et Z. Yakhini (2002). Discovering local structure in gene expression data: The order-preserving submatrix problem. *In Proceedings of the 6th International Conference on Computacional Biology (RECOMB'02)*, 49–57.

Besson, J. (2005). Découvertes de motifs pertinents pour l'analyse du transcriptome : Application à l'insulino-résistance. *Thèse de doctorat, INSA de Lyon*.

Bock, H. (1979). Simultaneous clustering of objects and variables. *In E., editor, Analyse des Données et Informatique*, 187–203.

Bock, H. (2003). Two-way clustering for contingency tables maximizing a dependence measure. *In Schader, M., Gaul, W., and Vichi, M., editors, Between Data Science and Applied Data Analysis*, 143–155.

Busygin, S., G. Jacobsen, et E. Kramer (2002). Double conjugated clustering applied to leukemia microarray data. *In Proceedings of the 2nd SIAM International Conference on Data Mining, Workshop on Clustering High Dimensional Data, Arlington Virginia, USA*, 143–155.

Califano, A., G. Stolovitzky, et Y. Tu (2000). Analysis of gene expression microarrays for phenotype classification. *In Proceedings of the 6th International Conference on Computacional Biology (RECOMB'02)*, 75–85.

Charrad, M. (2010). *Une approche générique pour l'analyse croisant contenu et usage des sites Web par des méthodes de bipartitionnement*. Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers, Paris, France.

Charrad, M., Y. Lechevallier, et M. B. Ahmed (2011). Simultaneous clustering : State of the art. *4th International Conference on Pattern Recognition and Machine Intelligence. S.O. Kuznetsov et al. (Eds.): PReMI 2011, LNCS 6744, pp. 370-375. Springer, Heidelberg, 2011*.

Charrad, M., Y. Lechevallier, G. Saporta, et M. B. Ahmed (2008). Le bipartitionnement : Etat de l'art sur les approches et les algorithmes. *Ecol'IA'08, Hammamet, Tunisie*.

Cheng, Y. et G. M. Church (2000). Biclustering of expression data. *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'00)*, 93–103.

Dhillon, I. (2001). Coclustering documents and words using bipartite spectral graph partitioning. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), California, USA*, 269–274.

Dhillon, I., S. Mallela, et D. Modha (2003). Information-theoretic co-clustering. *In ACM SIGKDD, Washington, DC,USA, ACM*, 89–98.

Getz, G., E. Levine, et E. Domany (2000). Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci. USA*.

Govaert, G. (1977). Algorithme de classification d'un tableau de contingence. *In First international symposium on data analysis and informatics*, 487–500.

Govaert, G. (1983). Classification croisée. *Thèse de doctorat d'état, Paris*, 463–473.

Govaert, G. et M. Nadif (2003). Clustering with block mixture models. *Pattern Recognition*, 463–473.

Govaert, G. et M. Nadif (2005). An em algorithm for the block mixture models. *IEEE Transactions on Pattern Analysis and machine intelligence 67*, 643–647.

Govaert, G. et M. Nadif (2009). Un modèle de mélange pour la classification croisée d'un tableau de données continues. *CAP'09, 11e conférence sur l'apprentissage artificiel, Hammamet : Tunisie (2009)*.

Hartigan, J. (1972). Direct clustering of a data matrix. *Journal of American Statistical Association, Vol.67, N.337 67*, 123–129.

Hartigan, J. (1975). Direct splitting. *Dans Clustering Algorithms. John Wiley and Sons, New York*, 251–277.

Kluger, Y., R. Basri, T. Joseph, et C. Gerstein (2003). Spectral biclustering of microarray data: coclustering genes and conditions. *In Genome Research, vol. 13*, 703–716.

Lazzeroni, L. et A. Owen. Plaid models for gene expression data. *Technical report, Stanford University*.

Madeira, S. C. et A. L. Oliveira (2004). Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 24–45.

Marchotorchino, F. (1987). Block seriation problems : A unified approach. *Applied Stochastic Models and Data Analysis*, 73–91.

Mechelen, I. V., H. Bock, et P. D. Boeck (2004). Two-mode clustering methods : a structured overview. *Statist. Methods Medical Res., 13(5)*, 363–394.

Mirkin, B. (1996). Mathematical classification and clustering. *Dordrecht: Kluwer*, 251–277.

Nadif, M. et G. Govaert (2005). Block clustering via the block gem and two-way em algorithms. *The third ACS/IEEE International conference on Computer Systems and Applications, (in CD ISBN 0-7803-8735-X) Cairo, Egypt, 03-06 january*, 463–473.

Robardet, C. (2002). Contribution à la classification non supervisée : proposition d'une méthode de bi-partitionnement. *thèse de doctorat, Université Claude Bernard - Lyon 1*, 463–473.

Tanay, A., R. Sharan, et R. Shamir (2002). Discovering statistically significant biclusters in gene expression data. *In Bioinformatics, volume 18 (Suppl. 1)*, 136–144,.

Tanay, A., R. Sharan, et R. Shamir (2004). Biclustering algorithms: A survey. *In Handbook of Computational Molecular Biology, Edited by Srinivas Aluru, Chapman*.

Vichi, M. (2000). Double k-means clustering for simultaneous of objects and variables. *In Borra, S. e. a., editor, Advances in Classification and Data Analysis*.

Vichi, M. et H. Kiers (2001). Factoriel k-means analysis for two-way data. *Computational Statistics and Data Analysis, Vol. 37, No.1*, 49–64.

Yang, J., H. Wang, W. Wang, et P. Yu (2003). Enhanced biclustering on expression data. *In: 3rd IEEE International Symposium on BioInformatics and BioEngineering (BIBE 2003), IEEE Computer Society, Los Alamitos, CA*, 321–327.

Zhang, C. T. C. L., M. Ramanathan, et A. Zhang (2001). Interrelated two-way clustering: An unsupervised approach for gene expression data analysis. *Proceedings of the 2nd IEEE International Symposium on Bioinformatics and Bioengineering*.

Summary

Simultaneous clustering has recently gained attention as a powerful tool that allows to circumvent some limitations of classical clustering approach. Simultaneous clustering, usually designated by biclustering, co-clustering, 2-way clustering or block clustering, performs clustering in the two dimensions simultaneously. A number of algorithms that perform simul-

taneous clustering on rows and columns of a matrix have been proposed to date. The goal of simultaneous clustering is to find sub-matrices, which are subgroups of rows and subgroups of columns that exhibit a high correlation. The current paper consider an acceleration of Croki2 algorithm proposed for contingecy table biclustering. We report results of our recent testing of accelerated Croki2 by the use of artificial datasets.