Co-clustering de données mixtes à base des modèles de mélange

Aichetou Bouchareb, Marc Boullé, Fabrice Rossi

Co-clustering de données mixtes à base des modèles de mélange

Aichetou Bouchareb*, Marc Boullé*, Fabrice Rossi**

*Orange Labs
prenom.nom@orange.com
**SAMM EA 4534 - Université Paris 1 Panthéon-Sorbonne
prenom.nom@univ-paris1.fr

Résumé. La classification croisée (co-clustering) est une technique non supervisée qui permet d'extraire la structure sous-jacente existante entre les lignes et les colonnes d'une table de données sous forme de blocs. Plusieurs approches ont été étudiées et ont démontré leur capacité à extraire ce type de structure dans une table de données continues, binaires ou de contingence. Cependant, peu de travaux ont traité le co-clustering des tables de données mixtes. Dans cet article, nous étendons l'utilisation du co-clustering par modèles à blocs latents au cas des données mixtes (variables continues et variables binaires). Nous évaluons l'efficacité de cette extension sur des données simulées et nous discutons ses limites potentielles.

1 Introduction

La classification croisée a pour objectif de réaliser une classification jointe des lignes et des colonnes d'un tableau de données. Proposée par Good (1965) puis par Hartigan (1975), la classification croisée est une extension de la classification simple (clustering) qui permet d'extraire la structure sous-jacente dans les données sous forme de groupes de lignes et groupes de colonnes. L'avantage de cette technique, par rapport à la classification simple, réside dans l'étude *simultanée (jointe)* des lignes et des colonnes qui permet d'extraire un maximum d'informations sur la dépendance entre elles. L'utilité du co-clustering réside dans sa capacité de créer des groupes facilement interprétables et dans sa capacité de réduction d'une grande table de données en une matrice significativement plus petite et ayant la même structure que les données originales. Le traitement de la matrice résumée permet d'étudier et de prendre des décisions sur les données originales tout en réduisant significativement les coûts de calcul en temps et en mémoire.

Depuis son introduction, plusieurs méthodes ont été développées pour effectuer une classification croisée (Bock (1979); Cheng et Church (2000); Dhillon et al. (2003); Xu et al. (2010)). Ces méthodes diffèrent principalement dans le type des données étudiées (continues, binaires ou de contingence), les hypothèses considérées, la méthode d'extraction utilisée et la forme souhaitée pour les résultats (classification stricte ou floue, hiérarchie, etc.). L'une des approches les plus connues est celle de la classification par modèles à blocs latents qui est basée sur des

modèles de mélanges où les classes des lignes et les classes des colonnes sont définies par des variables latentes à estimer (Govaert et Nadif (2003, 2007, 2008, 2010, 2013)). Ces modèles étendent au contexte de la classification croisée les mélanges de gaussiennes pour les données continues et ceux de Bernoulli pour les données binaires.

Des modèles à blocs latents ont ainsi été proposés et validés pour des données numériques, binaires, de contingence et catégorielles (Govaert et Nadif (2013)). Toutefois, à notre connaissance, ces modèles n'ont pas été appliqués aux données mixtes. En pratique, les données ne se présentent pas seulement sous forme continue ou binaire et l'extraction des informations nécessite le traitement des données mixtes. Sachant que la plupart des méthodes d'analyse de données sont conçues pour un type particulier des données en entrée, l'analyste se trouve obligé de passer par une phase de pré-traitement des données en les transformant sous une forme uni-type (souvent binaire) pour utiliser une approche appropriée, ou d'analyser ses données indépendamment par type et d'effectuer une interprétation conjointe des résultats. Le pré-traitement des données de types différents risque de faire perdre des informations importantes et le traitement indépendant risque de rendre difficile l'interprétation des résultats des méthodes reposant sur différents types de modélisation.

Les modèles de mélange ont été utilisés pour étudier des données mixtes dans le contexte de clustering par McParland et Gormley (2016) qui proposent un modèle à variables latentes suivant une loi gaussienne quel que soit le type des données (numérique, binaire, ordinales ou nominales) mais l'utilisation de ces modèles en classification croisée des donnés mixtes reste inexistante. Nous proposons d'étendre les modèles proposés dans Govaert et Nadif (2003, 2008) au cas des données mixtes (continues et binaires), en adoptant comme ces auteurs une approche d'estimation par maximum de vraisemblance.

Le reste de cet article est organisé comme suit : dans la section 2, nous commençons par définir les modèles à blocs latents et leur utilisation pour la classification croisée. Dans la section 3, nous présentons notre extension pour les données mixtes. La section 4 présente les résultats expérimentaux sur des données simulées et la section 5 les conclusions et perspectives.

2 Co-clustering par modèles à blocs latents

Considérons le tableau de données $\mathbf{x} = (x_{ij}, i \in I, j \in J)$ où I est un ensemble de n objets et J l'ensemble des d variables caractérisant les objets (représentés respectivement par les lignes et les colonnes de la matrice $\mathbf{x}$). L'objectif est d'obtenir une partition des lignes en g groupes et une partition des colonnes en m groupes, notées $\mathcal{Z}$ et $\mathcal{W}$, qui permettent, après avoir réordonné les lignes et les colonnes par groupe, de former des $blocs$ homogènes aux intersections de ces groupes. Nous supposons ici que le nombre de classes en lignes et de classes de colonnes g et m sont connus. Un élément x_{ij} appartient au bloc $B_{kl} = (I_k, J_l)$ si et seulement si la ligne $x_{i.}$ appartient au groupe I_k des lignes et la colonne $x_{.j}$ appartient au groupe J_l des colonnes. Les partitions des lignes et des colonnes sont représentées par une matrice binaire d'appartenance aux classes de lignes $\mathbf{z}$ et une matrice binaire d'appartenance aux classes des colonnes $\mathbf{w}$ où $z_{ik} = 1$ si et seulement si $x_{i.} \in I_k$ et $w_{jl} = 1$ si et seulement si $x_{.j} \in J_l$.

La vraisemblance du modèle à blocs latents s'écrit alors :

$$f(\mathbf{x};\theta) = \sum_{(\mathbf{z},\mathbf{w})\in\mathcal{Z}\times\mathcal{W}} p((\mathbf{z},\mathbf{w});\theta) f(\mathbf{x}|\mathbf{z},\mathbf{w};\theta)), \tag{1}$$

où $\mathcal{Z} \times \mathcal{W}$ représente l'ensemble de toutes les partitions possibles $\mathbf{z}$ de I et toutes les partitions possibles $\mathbf{w}$ de J qui vérifient les conditions ci-dessous et θ l'ensemble des paramètres inconnus du modèle.

Hypothèses : Le modèle à blocs latents (LBM) est basé sur 3 hypothèses principales

1. Il existe des partitions des lignes en g classes $\{I_1, \ldots, I_g\}$ et des partitions des colonnes en m classes $\{J_1, \ldots, J_m\}$ telles que chaque élément x_{ij} est le résultat d'une distribution de probabilité qui ne dépend que de sa classe en ligne et de sa classe en colonne. Ces partitions peuvent être représentées par des variables latentes à estimer.
2. Les appartenances aux classes lignes et aux classes colonnes sont indépendantes.
3. Connaissant les appartenances aux classes, les données observées sont indépendantes (indépendance conditionnelle au couple($\mathbf{z}$, $\mathbf{w}$)).

Sous ces hypothèses, la vraisemblance du modèle s'écrit alors :

$$f(\mathbf{x};\theta) = \sum_{(\mathbf{z},\mathbf{w})\in\mathcal{Z}\times\mathcal{W}} \prod_{ik} \pi_k^{z_{ik}} \prod_{jl} \rho_l^{w_{jl}} \prod_{ijkl} \varphi_{kl}(x_{ij};\alpha_{kl})^{z_{ik}w_{jl}},$$

et la log-vraisemblance est donnée par

$$L(\theta) = \log f(\mathbf{x};\theta) = \log \left(\sum_{(\mathbf{z},\mathbf{w})\in\mathcal{Z}\times\mathcal{W}} \prod_{ik} \pi_k^{z_{ik}} \prod_{jl} \rho_l^{w_{jl}} \prod_{ijkl} \varphi_{kl}(x_{ij};\alpha_{kl})^{z_{ik}w_{jl}} \right),$$

où les sommes et les produits sur i, j, k, l ont des limites de 1 à n, d, g, et m respectivement, π_k et ρ_l sont les proportions de la k$^{\text{ème}}$ classe des lignes et de la l$^{\text{ème}}$ classe des colonnes, α_{kl} est l'ensemble des paramètres du bloc B_{kl}. La vraisemblance φ_{kl} est celle d'une loi gaussienne pour les données continues et correspond une loi de Bernoulli pour les données binaires.

Pour une matrice $\mathbf{x}_{(n\times d)}$ et pour une partition en $g \times m$ classes, la somme sur $\mathcal{Z} \times \mathcal{W}$ nécessite au moins $g^n \times m^d$ opérations (Brault et Lomet (2015)) : il est donc impossible de calculer directement la log-vraisemblance en un temps raisonnable, ce qui empêche une application directe de l'algorithme EM classiquement utilisé pour les modèles de mélange. Govaert et Nadif (2008) utilisent donc une approximation variationnelle (et un algorithme VEM).

3 Contribution

Pour la suite, nous considérons une table des données mixtes $\mathbf{x} = (x_{ij}, i \in I, j \in J = J_c \cup J_d)$ où I est un ensemble de n objets décrits par des variables continues et binaires avec J_c l'ensemble des d_c variables continues et J_d l'ensemble des d_d variables binaires. L'objectif est d'obtenir une partition des lignes en g groupes, une partition des colonnes continues en m_c groupes et une partition des colonnes binaires en m_d groupes, notées $\mathcal{Z}$, $\mathcal{W}_c$ et $\mathcal{W}_d$ respectivement. Nous supposons que la partition des lignes, la partition des colonnes continues et la partition des colonnes binaires sont indépendantes. Ces partitions sont représentées par les matrices binaires de classification $\mathbf{z}$, $\mathbf{w}_c$, $\mathbf{w}_d$ et par les matrices de classification floue s, tc et td respectivement. De plus, conditionnellement à $\mathbf{w}_c$, $\mathbf{w}_d$ et $\mathbf{z}$, les $(x_{ij})_{\{i\in I, j\in J\}}$ sont

indépendantes et il existe un moyen de distinguer les colonnes continues des discrètes. Sous ces hypothèses, la vraisemblance du modèle génératif des données mixtes s'écrit :

$$f(\mathbf{x};\theta) = \sum_{(\mathbf{z},\mathbf{w}_c,\mathbf{w}_d)\in\mathcal{Z}\times(\mathcal{W}_c,\mathcal{W}_d)} \left(\prod_{ik} \pi_k^{z_{ik}} \prod_{j_c l_c} \rho_{l_c}^{wc_{j_c l_c}} \prod_{j_d l_d} \rho_{l_d}^{wd_{j_d l_d}} \right.$$
$$\left. \prod_{ij_c kl_c} \varphi^c_{kl_c}(x_{ij_c};\alpha_{kl_c})^{z_{ik} wc_{j_c l_c}} \prod_{ij_d kl_d} \varphi^d_{kl_d}(x_{ij_d};\alpha_{kl_d})^{z_{ik} wd_{j_d l_d}} \right)$$

Notons que les hypothèses conduisent à une simple combinaison des deux situations précédentes (binaire et numérique). Il n'y a donc pas de difficulté mathématique nouvelle par rapport au cas non mixte, mais plutôt des conséquences pratiques potentielles, induites par le couplage entre des distributions très différentes (par la classification des lignes) et par le caractère incommensurable des densités (variables continues) et des probabilités (variables binaires).

Nous optimisons la vraisemblance par un algorithme itératif de type VEM inspiré de Govaert et Nadif (2008) décrit ci-dessous.

3.1 Vraisemblance complétée

Nous dérivons tout d'abord l'expression de la log vraisemblance complétée, qui demande la connaissance des valeurs des variables latentes, ici $\mathbf{z}$, $\mathbf{w}_c$ et $\mathbf{w}_d$. On a

$$L_c(\mathbf{x},\mathbf{z},\mathbf{w}_c,\mathbf{w}_d;\theta) = \sum_{ik} z_{ik} \log \pi_k + \sum_{j_c l_c} wc_{j_c l_c} \log \rho_{l_c} + \sum_{j_d l_d} wd_{j_d l_d} \log \rho_{l_d}$$
$$+ \sum_{ij_c kl_c} z_{ik} wc_{j_c l_c} \log \varphi^c_{kl_c}(x_{ij_c};\alpha_{kl_c}) + \sum_{ij_d kl_d} z_{ik} wd_{j_d l_d} \log \varphi^d_{kl_d}(x_{ij_d};\alpha_{kl_d}),$$

où les sommes sur i, j_c, j_d, k, l_c, l_d ont des limites de 1 à n, d_c, d_d, g, m_c et m_d respectivement.

3.2 Approximation variationnelle

Dans les modèles à blocs latents, il n'est pas possible d'appliquer l'algorithme EM directement vu la dépendance entre $\mathbf{z}$ et $\mathbf{w}_c$ d'un côté et entre $\mathbf{z}$ et $\mathbf{w}_d$ de l'autre côté ce qui rend infaisable le calcul de la distribution jointe $p(\mathbf{z},\mathbf{w}_c,\mathbf{w}_d|\mathbf{x},\theta)$. On ne peut donc pas intégrer la log vraisemblance complétée par rapport à cette distribution.

Comme dans Govaert et Nadif (2008), nous utilisons une approximation variationnelle qui consiste à approcher les distributions conditionnelles des variables latentes sous une forme factorisable. Plus précisément, nous approchons $p(\mathbf{z},\mathbf{w}_c,\mathbf{w}_d|\mathbf{x},\theta)$ par le produit de distributions ajustables $q(\mathbf{z}|\mathbf{x},\theta)$, $q(\mathbf{w}_c|\mathbf{x},\theta)$ et $q(\mathbf{w}_d|\mathbf{x},\theta)$, de paramètres $s_{ik} = q(z_{ik}=1|\mathbf{x},\theta)$, $tc_{jl} = q(wc_{jl}=1|\mathbf{x},\theta)$ et $td_{jl} = q(wd_{jl}=1|\mathbf{x},\theta)$ respectivement.

On minore ainsi la vraisemblance complétée par le critère F_c suivant

$$F_c(s,tc,td,\theta) = \sum_{ik} s_{ik} \log \pi_k + \sum_{j_c l_c} tc_{j_c l_c} \log \rho_{l_c} + \sum_{j_d l_d} td_{j_d l_d} \log \rho_{l_d}$$
$$+ \sum_{ij_c kl_c} s_{ik} tc_{j_c l_c} \log \varphi^c_{kl_c}(x_{ij_c};\alpha_{kl_c}) + \sum_{ij_d kl_d} s_{ik} td_{j_d l_d} \log \varphi^d_{kl_d}(x_{ij_d};\alpha_{kl_d})$$
$$- \sum_{ik} s_{ik} \log s_{ik} - \sum_{j_c l_c} tc_{j_c l_c} \log tc_{j_c l_c} - \sum_{j_d l_d} td_{j_d l_d} \log td_{j_d l_d}.$$

3.3 Algorithme VEM

La maximisation de la borne inférieure F_c se fait, jusqu'à convergence à un ϵ près, en trois étapes :

- par rapport à s avec θ, tc et td fixés, ce qui revient à calculer

$$\hat{s}_{ik} \propto \pi_k \exp\left(\sum_{j_c l_c} tc_{j_c l_c} \log \varphi^c_{kl_c}(x_{ij_c}, \alpha_{kl_c})\right) \exp\left(\sum_{j_d l_d} td_{j_d l_d} \log \varphi^d_{kl_d}(x_{ij_d}, \alpha_{kl_d})\right) \tag{2}$$

- par rapport à tc et td avec s et θ fixés, ce qui revient à calculer

$$\begin{aligned} \hat{tc}_{j_c l_c} &\propto \rho_{l_c} \exp\left(\sum_{ik} s_{ik} \log \varphi^c_{kl_c}(x_{ij_c}, \alpha_{kl_c})\right) \\ \text{et } \hat{td}_{j_d l_d} &\propto \rho_{l_d} \exp\left(\sum_{ik} s_{ik} \log \varphi^d_{kl_d}(x_{ij_d}, \alpha_{kl_d})\right), \end{aligned} \tag{3}$$

avec : $\sum_k s_{ik} = \sum_{l_c} tc_{jl_c} = \sum_{l_d} td_{jl_d} = 1$.

— par rapport à θ : ce qui revient à calculer les proportions des classes et leurs paramètres

$$\begin{aligned} \hat{\pi}_k = \frac{\sum_i \hat{s}_{ik}}{n};\ \hat{\rho}_{l_c} = \frac{\sum_{j_c} \hat{tc}_{jl_c}}{d_c};\ \hat{\rho}_{l_d} = \frac{\sum_{j_d} \hat{td}_{jl_d}}{d_d};\ \hat{\mu}_{kl_c} = \frac{\sum_{ij_c} \hat{s}_{ik} \hat{tc}_{j_c l_c} x_{ij_c}}{\sum_i \hat{s}_{ik} \sum_{j_c} \hat{tc}_{j_c l_c}}; \\ \hat{\sigma}^2_{kl_c} = \frac{\sum_{ij_c} \hat{s}_{ik} \hat{tc}_{j_c l_c} (x_{ij_c} - \hat{\mu}_{kl_c})^2}{\sum_i \hat{s}_{ik} \sum_{j_c} \hat{tc}_{j_c l_c}} \text{ et } \hat{\alpha}_{kl_d} = \frac{\sum_{ij_d} \hat{s}_{ik} \hat{td}_{j_d l_d} x_{ij_d}}{\sum_i \hat{s}_{ik} \sum_{j_d} \hat{td}_{j_d l_d}} \end{aligned} \tag{4}$$

Dans notre implémentation (algorithme 1), nous avons choisi $\epsilon = 10^{-5}$ pour les boucles intérieures, $\epsilon = 10^{-10}$ pour la boucle extérieure et nous normalisons $\hat{s}$, $\hat{tc}$ et $\hat{td}$, après calcul, en prenant les valeurs relatives : $\hat{s}_{ik} \leftarrow \frac{\hat{s}_{ik}}{\sum_h \hat{s}_{ih}}$ et de même pour $\hat{tc}$ et $\hat{td}$.

4 Expérimentations

Il est difficile d'évaluer les méthodes par modèles de mélange sur des données réelles dont la distribution sous-jacente est inconnue. Dans cette section nous présentons les expériences réalisées sur des jeux de données simulées. La première expérience a pour objectif de valider notre implémentation sur des données uni-type et de vérifier l'apport de l'approche. La deuxième expérience nous permettra d'étudier l'influence de plusieurs paramètres tels que le nombre de blocs, la taille de la matrice et le niveau de bruit.

4.1 Première expérience

4.1.1 Le jeu de données

Notre premier jeu de données simulé est une matrice de $g = 4$ classes de lignes, $m_c = 2$ classes de colonnes continues et $m_d = 2$ classes de colonnes binaires. Prises individuellement, les parties continues et binaires permettent de distinguer seulement deux classes de lignes

Algorithm 1 : Latent Block VEM étendu

Require: $\mathbf{x}$, g, m_c, m_d
  iteration c$\leftarrow$0
  Initialiser : $s = c^c$, $td = tc^c, td = td^c$ aléatoirement et calculer $\theta = \theta^c$ (Équation (4))
  while $c \leq maxITER$ **and** Unstable($Criterion$) **do**
    t$\leftarrow$0, $s^t \leftarrow s^c, tc \leftarrow tc^c, td \leftarrow td^c, \theta^t \leftarrow \theta^c$
    while $t \leq InnerMaxIter$ **and** Unstable($Criterion$) **do**
      Pour tout $i = 1 : n$ et $k = 1 : g$, calculer s_{ik}^{t+1} : Équation (2)
      Pour tout $k = 1 : g$, $l_c = 1 : m_c$ et $l_d = 1 : m_d$, calculer $\pi_k^{t+1}, \mu_{kl_c}^{t+1}, \sigma_{kl_c}^{t+1}$ et $\alpha_{kl_d}^{t+1}$: Équation (4)
      $Criterion \leftarrow F_c(s^{t+1}, tc, td, \theta^{t+1})$
      $t \leftarrow t + 1$
    end while
    $s \leftarrow s^{c+1} \leftarrow s^{t-1}$, $\theta \leftarrow \theta^{c+1} \leftarrow \theta^{t-1}$
    $t \leftarrow 0$
    while $t \leq InnerMaxIter$ **and** Unstable($Criterion$) **do**
      Pour tout $j_c = 1 : d_c$, $j_d = 1 : d_d$, $l_c = 1 : m_c$ et $l_d = 1 : m_d$, calculer $tc_{j_c l_c}^{t+1}$ et $td_{j_d l_d}^{t+1}$: Équation (3)
      Pour tout $k = 1 : g$, $l_c = 1 : m_c$ et $l_d = 1 : m_d$, calculer ρ_c^{t+1}, ρ_d^{t+1}, $\mu_{kl_c}^{t+1}$, $\sigma_{kl_c}^{t+1}$ et $\alpha_{kl_d}^{t+1}$: Équation (4)
      $Criterion \leftarrow F_c(s, tc^{t+1}, td^{t+1}, \theta^{t+1})$
      $t \leftarrow t + 1$
    end while
    $tc \leftarrow tc^{c+1} \leftarrow tc^{t-1}$, $td \leftarrow td^{c+1} \leftarrow td^{t-1}$, $\theta \leftarrow \theta^{c+1} \leftarrow \theta^{t-1}$
    $Criterion \leftarrow F_c(s, tc, td, \theta)$
    $c \leftarrow c + 1$
  end while
Ensure: (s, tc, td, θ)

mais conjointement, on espère trouver les quatre classes de lignes. Dans ce jeu de données, nous étudions l'influence de :

La taille de la matrice des données : le nombre de lignes, de colonnes continues et de colonnes binaires. Nous considérons les tailles de matrice représentées par 25, 50, 100, 200 et 400 lignes et colonnes de chaque type (les matrices mixtes résultantes ont donc 25×50, 50×100, 100×200, 200×400 et 400×800 éléments).

Le niveau de confusion entre les mélanges où nous considérons trois niveaux : *Faible* (moyennes des gaussiennes $\mu \in \{\mu_1 = 1, \mu_2 = 2\}$, écarts types des gaussiennes $\sigma = 0.25$ et paramètres de Bernoulli $\alpha \in \{\alpha_1 = 0.2, \alpha_2 = 0.8\}$), *Moyen* ($\mu \in \{\mu_1, \mu_2\}$, $\sigma = 0.5$ et $\alpha \in \{\alpha_1 = 0.3, \alpha_2 = 0.7\}$) et *Élevé* ($\mu \in \{\mu_1, \mu_2\}$, $\sigma = 1$ et $\alpha \in \{\alpha_1 = 0.4, \alpha_2 = 0.6\}$).

Le type de configuration : lors de nos expériences nous avons remarqué que l'algorithme a parfois du mal à retrouver la structure dans le cas de blocs carrés et dont les marginales des paramètres sont égales. Nous considérons donc deux configurations nommées *asymétrie* où les marginales des paramètres des classes lignes et des classes colonnes sont différentes et

symétrie où les marginales sont identiques. La spécification des paramètres des blocs continus et binaires par type de configuration est détaillée dans Table 1. Il est à noter que les mélanges de gaussiennes (colonnes Jc_1 et Jc_2) permettent de distinguer deux classes de ligne en associant $\{I_1$ et $I_3\}$ d'une part et $\{I_2$ et $I_4\}$ d'autre part. De même, les paramètres de Bernoulli (colonnes Jd_1 et Jd_2) permettent de distinguer deux classes de lignes en associant $\{I_1$ et $I_2\}$ d'une part et $\{I_3$ et $I_4\}$ d'autre part. En étudiant les données mixtes conjointement, on s'attend à distinguer quatre classes de lignes.

asymétrie

μ et α	Jc_1	Jc_2	Jd_1	Jd_2
I_1	μ_2	μ_1	α_2	α_1
I_2	μ_2	μ_2	α_2	α_1
I_3	μ_2	μ_1	α_2	α_2
I_4	μ_2	μ_2	α_2	α_2

symétrie

μ et α	Jc_1	Jc_2	Jd_1	Jd_2
I_1	μ_1	μ_2	α_1	α_2
I_2	μ_2	μ_1	α_1	α_2
I_3	μ_1	μ_2	α_2	α_1
I_4	μ_2	μ_1	α_2	α_1

TAB. 1 – *La vraie spécification des blocs dans les configurations asymétrie et symétrie.*

Nos expérimentations sont effectuées en deux étapes : appliquer l'algorithme de coclustering aux données continues seules et binaires seules pour valider notre implémentation et appliquer ensuite l'algorithme sur l'ensemble des données mixtes. Avec 5 tailles de matrice, 3 niveaux de confusion, 2 types de configuration, un co-clustering appliqué à la partie continue, la partie binaire ou à l'ensemble des variables mixtes, au total 90 expériences ont ainsi été effectuées.

Connaissant les vraies classes des données, nous mesurons la performance d'une classification en blocs par l'Indice de Rand Ajusté (ARI), qui permet de mesurer l'écart entre la partition retrouvée par les méthodes de co-clustering et la vraie partition. Nous nous focalisons sur la capacité de notre approche à potentiellement mieux identifier les classes de lignes en utilisant toutes les variables de types mixtes, et nous collectons à cet effet les ARI lignes dans les trois cas de figure : ARI observé dans l'étude des variables continues (ARI_cont), binaires (ARI_disc) ou mixtes (ARI_mix). Pour chaque type de jeux de données, nous générons 5 échantillons de données selon les paramètres de la configuration et nous présentons les résultats sous la forme de diagrammes en violon. Les diagrammes en violon (Hintze et Nelson (1998)) permettent de combiner les avantages des boîtes à moustaches et des estimateurs de densité de probabilité des différentes valeurs, permettant ainsi une meilleure visualisation de la variabilité des résultats.

4.1.2 Validation de l'implémentation

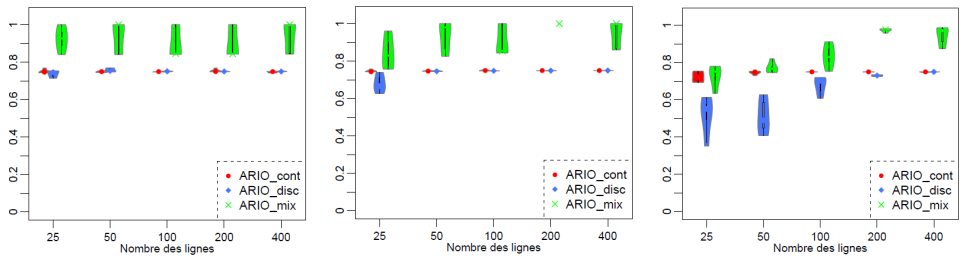
Pour valider notre implémentation, nous avons appliqué l'algorithme aux données continues seules et aux données binaires seules en comparant nos résultats avec ceux du package blockcluster (Bhatia et al. (2014)).

Dans un premier temps, nous avons pu vérifier que notre implémentation obtient au moins les mêmes performances en ARI que ceux de blockcluster. En appliquant l'algorithme sur des données uni-type, nous avons néanmoins observé quelques problèmes dans l'optimisation du critère. En premier, l'algorithme converge rapidement vers un optimum local et très souvent cet optimum correspond à une seule classe de lignes et une seule classe de colonnes. Ce point

peut être amélioré en imposant un nombre minimal d'itérations (le paramètre c dans l'algorithme1) ce qui permet d'améliorer significativement la qualité des résultats d'optimisation. Dans la configuration *symétrie* où les marginales des paramètres sont égales, le problème de séparabilité des classes devient intrinsèquement plus difficile et l'algorithme a du mal à sortir de la zone de l'optimum local correspondant à une classe de lignes et une classe de colonnes dans lequel il tombe dès la première itération. Pour remédier à ce problème, nous commençons l'algorithme par des petits pas dans le calcul des appartenances aux classes (s, tc et td) sans que le critère se stabilise, puis après les premières itérations de la phase initiale, on itère jusqu'à stabilisation. Cette stratégie permet de trouver une meilleure solution dans le cas des données binaires mais ne permet pas d'amélioration notable dans le cas continu. Dans la configuration symétrie, les résultats obtenus tant avec notre implémentation qu'avec le package blockcluster sont de faible qualité, avec une variance importante due aux problèmes d'optimisation. Dans la suite de l'article, nous ne présentons pas les résultats portant sur cette configuration, d'une part par manque de place, d'autre part parce que la faible qualité des résultats n'a pas permis d'observer des variations de comportement significatives dans les expériences effectuées.

4.1.3 Apport du co-clustering de données mixtes

Nous vérifions ici si la méthode exploitant les données mixtes arrive effectivement à identifier les quatre clusters de lignes. Comme attendu, les méthodes n'exploitant que les données continues ou binaires séparément ne peuvent pas identifier les quatre clusters correctement, et leur ARI ne dépasse jamais 0.7. En revanche, nous remarquons une amélioration importante et significative des ARI quand les données continues et binaires sont utilisées conjointement (cf. figure 1).



(a) Confusion faible (b) Confusion moyenne (c) Confusion élevée

FIG. 1 – *Première expérience (asymétrie) : ARI des lignes en continu, binaire et mixte.*

Rappelons que 5 échantillons sont générés pour chaque configuration de données et que les ARI sont calculés pour chaque échantillon. Les diagrammes en violon permettent de voir des statistiques comme la moyenne, la médiane et l'étendue mais aussi la distribution des valeurs. Par exemple, dans la figure 1a, les ARI de la partie continue (en rouge) de tous les échantillons de taille 25 ont la même valeur de 0.75 environ et ont donc une étendue et une variance nulles. Dans l'étude du cas mixte, le diagramme en violon montre (en vert) une concentration plus importante autour de 0.9 pour la matrice de taille 25 et autour de 1 pour la matrice de taille 50.

Il apparaît clairement que quelle que soit la configuration et quelle que soit la taille de la matrice, la prise en compte des données mixtes améliore l'ARI de façon significative.

Influence de la taille de la matrice. La quantité de données disponibles augmente avec la taille de la matrice, ce qui facile la convergence des algorithmes vers les vraies distributions sous-jacentes. Cette tendance est observée systématiquement sur la figure 1, notamment dans les cas binaire et mixte en cas de confusion élevée.

Influence du niveau de confusion. Quand le niveau de bruit augmente, il devient plus difficile de retrouver la bonne partition des lignes. Cet effet est visible en particulier sur la figure 1c, où le niveau élevé de confusion rend plus difficile la séparation des classes de lignes dans les cas binaire ou mixte, surtout quand les matrices sont petites.

Pour résumer. L'utilisation conjointe des variables continues et binaires permet d'identifier une structure en quatre clusters de lignes, ce qui dans le jeu de donné étudié n'est pas possible avec les variables continues seules ou binaires seules. Les résultats obtenus avec le co-clustering des données mixtes sont d'autant meilleurs que le niveau de bruit est faible, et que la taille des matrices est grande.

4.2 Deuxième expérience

4.2.1 Le jeu de données

Nous ne considérons ici que des configurations asymétriques, avec des marginales de valeurs des paramètres différentes sur chaque cluster de lignes ou de colonnes, afin d'utiliser les algorithmes de co-clustering là où ils sont performants (cf. section 4.1). Pour ce jeu de données, nous avons choisi une configuration où les clusters de lignes peuvent être identifiés dans tous les cas de figure : en utilisant les variables continues seules, binaires seules ou mixtes. Pour étudier l'influence du nombre de blocs sur le co-clustering du mixte, notre deuxième jeu de données est généré en fonction des paramètres suivants :

Le nombre de blocs : nous considérons 3 configurations de partitions $g \times (m_c + m_d)$ de la matrice initiale : $2 \times (2 + 2)$, $3 \times (3 + 3)$ et $4 \times (4 + 4)$.

La taille de la matrice des données : nous considérons les tailles de 25, 50, 100, 200 et 400 lignes et colonnes de chaque type.

Le niveau de confusion entre les mélanges où nous considérons trois niveaux : *Faible* (moyennes des gaussiennes $\mu \in \{p_1 = 1, p_2 = 2\}$, écart type des gaussiennes $\sigma = 0.25$ et paramètre de Bernoulli $\alpha \in \{p_1 = 0.2, p_2 = 0.8\}$), *Moyen* ($\mu \in \{p_1, p_2\}$, $\sigma = 0.5$ et $\alpha \in \{p_1 = 0.3, p_2 = 0.7\}$) et *Élevé* ($\mu \in \{p_1, p_2\}$, $\sigma = 1$ et $\alpha \in \{p_1 = 0.4, p_2 = 0.6\}$).

Les types de configuration utilisés sont présentés dans la table 2.

Avec 3 tailles de blocs, 5 tailles de matrice, 3 niveaux de confusions, un co-clustering appliqué à la partie continue, la partie binaire ou à l'ensemble des variables mixtes, au total 135 expériences ont ainsi été effectuées. Comme dans la première expérience, nous générons 5 échantillons de données selon les paramètres de chaque configuration et nous présentons les résultats d'ARI ligne sous la forme de diagrammes en violon.

4.2.2 Les résultats du co-clustering

La figure 2 présente les ARI des lignes du co-clustering des variables continues, binaires et mixtes. Pour des raisons de place, nous ne montrons que les configurations $2 \times (2 + 2)$

asymétrie

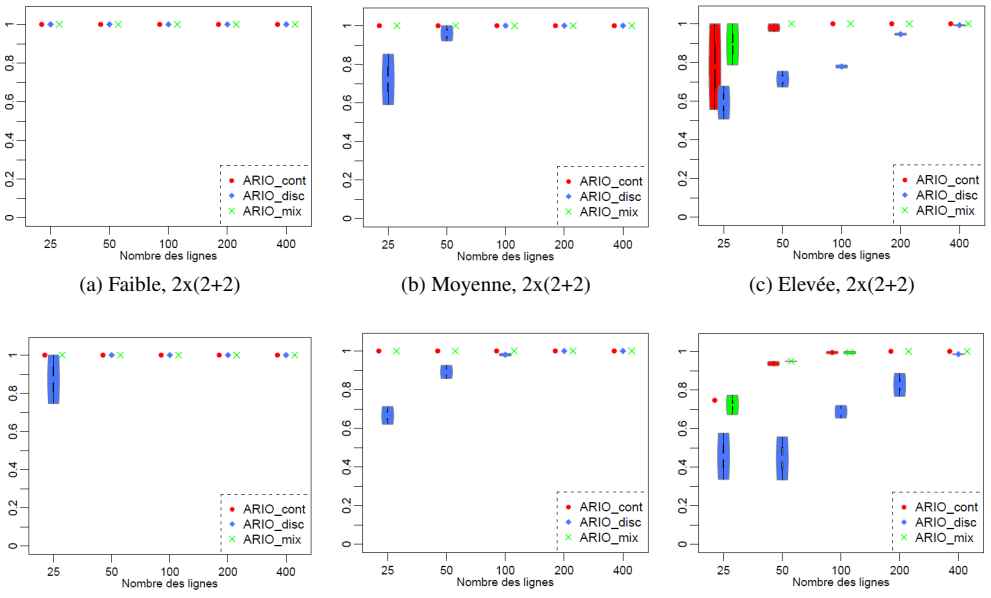
μ ou α	J_1	J_2
I_1	p_1	p_1
I_2	p_1	p_2

μ ou α	J_1	J_2	J_3
I_1	p_1	p_2	p_1
I_2	p_1	p_2	p_2
I_3	p_1	p_1	p_1

μ ou α	J_1	J_2	J_3	J_4
I_1	p_2	p_1	p_2	p_1
I_2	p_2	p_1	p_2	p_2
I_3	p_2	p_2	p_2	p_2
I_4	p_2	p_1	p_1	p_1

TAB. 2 – *La vraie spécification des blocs dans la configuration asymétrie pour* $2 \times (2+2)$, $3 \times (3+3)$ *et* $4 \times (4+4)$ *blocs.*

et $3 \times (3 + 3)$, qui illustrent bien l'influence de l'augmentation du nombre de blocs. Les configurations comportant le plus de blocs, $4 \times (4+4)$ non visualisées ici, confirment le même type d'impact quand le nombre de blocs augmente.



(a) Faible, 2x(2+2) (b) Moyenne, 2x(2+2) (c) Elevée, 2x(2+2)

(d) Faible, 3x(3+3) (e) Moyenne, 3x(3+3) (f) Elevée, 3x(3+3)

FIG. 2 – *Deuxième expérience : ARI des lignes en continu, binaire et mixte.*

Comme dans la première expérience, l'utilisation des données mixtes améliore nettement les performances du co-clustering, alors même qu'ici, les variable continues seules ou binaires seules permettaient en théorie de retrouver les clusters de lignes. La première constatation est que la partie binaire des données est sensible à la taille de la matrice, au nombre de blocs et au niveau de confusions alors que la partie continue est généralement plus stable et n'est influencée que par le niveau de confusion.

Influence du nombre de blocs. Nous constatons sur la figure 2 que plus le nombre de blocs est grand, plus il est difficile de séparer les classes. Cet effet est observé en particulier dans le cas des variables binaires seules, où la variabilité des résultats est plus grande lorsque le nombre de blocs est important. Cette variabilité est moins présente dans le cas des donnés continues et encore moins dans le cas mixte.

Influence de la taille de la matrice. La partie continue est bien séparable quelle que soit la taille de la matrice alors que les blocs discrets sont moins séparables pour les petites matrices. La meilleure séparation des blocs mixtes est constatée dans les matrices moyennes à grandes quel que soit le nombre de blocs et quel que soit le niveau de confusion.

Influence du niveau de confusion. Dans nos expériences, on retrouve le comportement attendu par rapport au niveau de confusion dans les mélanges. Plus le niveau de confusion est élevé, plus il est difficile de classer correctement les lignes, en particulier dans le cas des petites matrices (partie gauche des figures 2f et 2c). Par contre, même avec un niveau de confusion élevé, la qualité des partitions des lignes s'améliore avec la taille des matrices.

Pour résumer. L'utilisation conjointe des variables continues et binaires permet d'améliorer la qualité des clusters de lignes obtenus avec le co-clustering, même dans le cas où ces clusters étaient identifiables avec les données continues seules ou binaires seules. Les résultats obtenus avec le co-clustering des données mixtes sont d'autant meilleurs que le niveau de bruit est faible, que le nombre de blocs est faible et que la taille des matrices est grande.

5 Conclusion et discussion

Dans cet article, nous avons présenté une extension du modèle à blocs latents pour le co-clustering des données mixtes. Les expériences montrent une amélioration systématique de la qualité des résultats obtenus, en utilisant les données continue et binaires conjointement plutôt que séparément. Bien que nous n'ayons montré que les résultats de classification des lignes, le co-clustering dans le cas mixte améliore également nettement la qualité de la classification des colonnes. Lors de nos expérimentations, nous avons remarqué que pour des jeux de données avec des distributions marginales égales, tant notre algorithme que les méthodes de l'état de l'art peinent à retrouver la structure en blocs des données, en restant figés dans des optimaux locaux en sortie des étapes d'initialisation. Ce problème peut être limitant pour l'utilisation en pratique de ce type d'approche, pour l'analyse exploratoire de données dont on ne connait pas la vraie distribution sous-jacente. Lors de travaux futurs, nous viserons à améliorer la résolution de ce problème algorithmique, à étendre l'approche au cas de données catégorielles au delà des données binaires, et à étudier des solutions de régularisation de type BIC pour retrouver automatiquement le nombre de clusters de ligne et de colonnes.

Références

Bhatia, P., S. Iovleff, et G. Govaert (2014). blockcluster : An r package for model based co-clustering. working paper or preprint : `https://hal.inria.fr/hal-01093554`.

Bock, H. (1979). Simultaneous clustering of objects and variables. In *E. Diday (ed) Analyse des données et Informatique*, pp. 187–203. INRIA.

Brault, V. et A. Lomet (2015). Revue des méthodes pour la classification jointe des lignes et des colonnes d'un tableau. *Journal de la Société Française de Statistique 156*(3), 27–51.

Cheng, Y. et G. M. Church (2000). Biclustering of expression data. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, Volume 8, pp. 93–103. AAAI Press.

Dhillon, I. S., S. Mallela, et D. S. Modha (2003). Information-theoretic co-clustering. In *Proceedings of the ninth international conference on Knowledge discovery and data mining*, pp. 89–98. ACM Press.

Good, I. J. (1965). Categorization of classification. In *Mathematics and Computer Science in Biology and Medicine*, pp. 115–125. Her Majesty's Stationery Office, London.

Govaert et M. Nadif (2010). Latent block model for contingency table. *Communications in Statistics, Theory and Methods 39*(3), 416 – 425.

Govaert, G. et M. Nadif (2003). Clustering with block mixture models. *Pattern Recognition 36*(2), 463–473.

Govaert, G. et M. Nadif (2007). Clustering of contingency table and mixture model. *European Journal of Operational Research 183*(3), 1055–1066.

Govaert, G. et M. Nadif (2008). Block clustering with Bernoulli mixture models : Comparison of different approaches. *Computational Statistics and Data Analysis 52*(6), 3233–3245.

Govaert, G. et M. Nadif (2013). *Co-Clustering*. ISTE Ltd and John Wiley & Sons Inc.

Hartigan, J. A. (1975). *Clustering Algorithms*. New York, NY, USA : John Wiley & Sons, Inc.

Hintze, J. L. et R. D. Nelson (1998). Violin plots : A box plot-density trace synergism. *The American Statistician 52*(2), 181–184.

McParland, D. et I. C. Gormley (2016). Model based clustering for mixed data : clustmd. *Advances in Data Analysis and Classification. Springer 10*(2), 155–169.

Xu, G., Y. Zong, P. Dolog, et Y. Zhang (2010). *Co-clustering Analysis of Weblogs Using Bipartite Spectral Projection Approach*, pp. 398–407. Springer Berlin Heidelberg.

Summary

Co-clustering is a data mining technique used to extract the underlying block structure between the rows and columns of a data matrix. Many approches have been studied and have shown their capacity to extract such structures in continuous, binary or contingency tables. However, very little work has been done to perform co-clustering on mixed type data. In this article, we extend the use of latent bloc models to co-clustering in the case of mixed data (continuous and binary variables). We then evaluate the effectiveness of our extention on simulated data and we discuss its potential limits.