ANR ClasSel

Modèles pour la classification croisée des données continues

Livrable 1.2

Table des matières

1 Introduction

La classification est l'organisation de données en plusieurs groupes suivant leurs similitudes. Chaque classe obtenue se compose d'éléments ayant des ressemblances et des différences avec ceux des autres classes. La classification a pour but de réduire la taille de données en les résumant. On en obtient alors une simplification en négligeant certains de leurs détails.

Il existe deux familles de méthodes pour la classification : la classification supervisée et la classification non supervisée. Dans le premier cas, on connaît déjà la classe des données et on veut affecter une nouvelle observation à une classe. Dans le second cas, aussi appelé *clustering* ou *classification automatique*, l'objectif est de trouver les classes de la population et d'affecter l'étiquette de la classe correspondante à chaque donnée. De la même façon, on organise le *clustering* en deux catégories : la classification hiérarchique et le partitionnement. La classification croisée entre dans cette dernière catégorie. Il s'agit d'une méthode de partitionnement visant à organiser en classes un tableau. Son étude commence dans les années 1970 avec Fisher, Anderberg et Hartignan ; et, dans la littérature, on lui prête d'autres noms comme « co-clustering » ou « block clustering ».

Cette méthode prend son sens avec l'arrivée de plus en plus massive de grands tableaux de données dans tous les domaines des activités humaines. En effet, cette méthode générique peut s'appliquer à de nombreux domaines. Par exemple, en marketing, on s'intéresse à l'analyse des données Netflix (location de DVD aux Etats-Unis) dans le but de les organiser en blocs d'individus et de films similaires. De même, en génomie, on organise les données génomiques (expressions de gènes) afin de regrouper des gènes ayant des similitudes et des structures anatomiques Manjunatha et al. (2007). On trouve aussi d'autres exemples d'application comme en écologie Miklós et al. (2005) ou bien encore en imagerie...

La classification croisée s'applique à des tableaux dont on souhaite organiser simultanément les lignes et les colonnes. Elle peut s'appliquer à des tableaux de type binaires, de contingence ou bien encore à des tableaux de données continues Govaert (1995). En 2009, (Govaert and Nadif, 2009) propose une nouvelle approche pour la classification croisée de données continues. Il propose un modèle de « block clustering » dont les partitions recherchées sont des variables latentes. La classification croisée est alors vue comme une méthode de classification automatique visant à organiser simultanément l'ensemble des lignes et des colonnes d'un tableau $\mathbf{x}$ en blocs homogènes. Son objectif est de définir un couple de partitions $(\mathbf{z}, \mathbf{w})$ où $\mathbf{z}$ est la partition de l'ensemble des lignes I en g classes et $\mathbf{w}$, la partition de l'ensemble des colonnes J en m classes.

La classification croisée pose un problème fondamental du critère de classification et du nombre de classes. Le modèle « block latent » se place dans un cadre statistique avec l'estimation de densité permettant ainsi de proposer des critères statistiques de sélection de modèle. En effet, il existe des critères de vraisemblance pénalisée tels que BIC (Bayesian Information criterion), AIC (Akaike Information criterion), ICL (Integrated Completed Likelihood), qui proposent une sélection de modèles en faisant un compromis entre la qualité d'ajustement et la complexité du modèle. Néanmoins, ils dépendent de la taille de l'échantillon ce qui pose un problème en classification croisée puisque la taille de l'échantillon dépend du nombre de lignes et du nombre de colonnes. Il s'agit donc étudier de manière théorique et expérimentale des techniques de sélection de modèles en

se basant soit sur des critères existants soit en en proposant de nouveaux.

L'objectif de ce document est de définir un plan d'expérience visant à être utilisé lors de l'étude de critères de sélection de modèles. En effet, il est important, avant de commencer tout travail, de poser l'ensemble des hypothèses nécessaires à l'intégration de résultats dans une méthode. Dans ce but, on cherche à contrôler les simulations des jeux de données issues d'un modèle de référence (ici, le modèle de mélange pour la classification croisée). On obtient ainsi une illustration (et/ou validation) des résultats théoriques par des exemples pertinents répondant aux hypothèses de travail.

Pour cela, on établit un protocole de simulation d'un jeu de données (utilisées pour les exemples d'application) suivant le pourcentage de mal classés. Ainsi, on étudiera le comportement de l'algorithme de classification dans les cas où les classes des données sont très séparées soit environ 5% de mal classés, moyennement séparées (12%) et peu séparées (20%). Il s'agit donc de savoir comment simuler un jeu de données correspondant aux hypothèses de travail et dont on contrôle le pourcentage de mal classés.

Notations

Le tableau

I, i, n	ensemble des lignes
J, j, d	ensemble des colonnes
$\mathbf{x}, x_i, x_j, x_{ij}$	tableau de données

Les classes

g, k	nombre de classes lignes
m, l	nombre de classes colonnes

La partition ligne

$\mathbf{z} = (z_{11}, ..., z_{ng})$	partition de l'ensemble des lignes
$z_{ik} = 1$	si i appartient à la classe k et 0 sinon
$z_k = \sum_i z_{ik}$	le nombre de lignes appartenant à la classe k
Z	l'ensemble de toutes les partitions possibles de $\mathbf{z}$

La partition colonnes

$\mathbf{w} = (w_{11}, ..., w_{dm})$	partition de l'ensemble des lignes
$w_{jl} = 1$	si j appartient à la classe l et 0 sinon
$w_l = \sum_j z_{jl}$	le nombre de lignes appartenant à la classe l
W	l'ensemble de toutes les partitions possibles de $\mathbf{w}$

Les paramètres

$\boldsymbol{\pi} = (\pi_1, ..., \pi_g)$	la probabilité d'appartenance à une classe ligne
$\boldsymbol{\rho} = (\rho_1, ..., \rho_m)$	la probabilité d'appartenance à une classe colonne
$\boldsymbol{\alpha}$	ensemble des paramètres du modèle
$\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\alpha})$	

Les fonctions

$\mathbf{h}(\mathbf{x})$	le classifieur qui associe à chaque donnée du tableau une classe ligne et colonne
$\mathbf{L}((\mathbf{z}, \mathbf{w}), (\mathbf{z'}, \mathbf{w'}))$	la fonction coût mesurant la perte lorsqu'on choisit la partition $(\mathbf{z'}, \mathbf{w'})$ alors que c'est $(\mathbf{z}, \mathbf{w})$

2 Le modèle

On considère un tableau $\mathbf{x}_{ij}$ de taille $n \times d$ et dont les lignes et les colonnes ont un comportement similaire. Les données sont continues et suivent un modèle de mélange, souvent utilisé en classification automatique. Les unités statistiques sont soit les lignes, soit les variables (les deux ayant les mêmes propriétés).

Le modèle de bloc latent est une extension du modèle de mélange pour la classification croisée. On suppose que chaque ligne et chaque colonne sont des variables aléatoires suivant une loi gaussienne. Les partitions lignes et colonnes, $(\mathbf{z}, \mathbf{w})$, sont des variables aléatoires (latentes) du modèle de mélange devenant ainsi le modèle de blocs latents. Ce modèle a été proposé par (Govaert and Nadif, 2009).

2.1 Modèle de bloc latent pour la classification croisée, définition

On est la recherche d'un couple de partitions $(\mathbf{z}, \mathbf{w})$ de l'ensemble des lignes et colonnes d'un tableau $\mathbf{x}$. Le but est d'obtenir après organisation du tableau suivant les partitions des blocs homogènes. On peut alors en proposer une réduction via critère statistique comme la moyenne par exemple. Cette recherche de partitions peut être vue comme une approximation d'estimations de modèle de probabilité. Il est donc naturel de définir la distribution du modèle. La densité du modèle de blocs latents, munie de l'hypothèse d'indépendance des partitions lignes et colonnes, est définie par :

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

L'indépendance de $\mathbf{z}$ et $\mathbf{w}$ a priori n'implique pas une indépendance a posteriori. En effet, conditionnées à $\mathbf{x}$, $\mathbf{z}$ et $\mathbf{w}$ ne sont pas indépendantes. En revanche, les x_{ij} sont supposés indépendants pour z_i et w_j fixés. On peut alors réécrire cette densité sous la forme :

$$f(\mathbf{x}|\mathbf{z},\mathbf{w},\boldsymbol{\theta}) = \prod_{i,j} f_{z_i,w_j}(x_{ij};\boldsymbol{\alpha}) \tag{2}$$

$$f(\mathbf{x};\boldsymbol{\theta}) = \sum_{(\mathbf{z},\mathbf{w})\in Z\times W} \prod_i \pi_{z_i} \prod_j \rho_{w_j} \prod_{i,j} f_{z_i,w_j}(x_{ij};\boldsymbol{\alpha}) \tag{3}$$

Où $f_{z_i,w_j}(x|\boldsymbol{\alpha})$ est la fonction de probabilité d'un ensemble défini sur $\mathbb{R}$.

De ce modèle, on génère aléatoirement des données. Dans un premier temps, on simule une partition ligne in g classes suivant une multinomiale de paramètres $\boldsymbol{\pi}$. On génère de façon similaire mais avec les paramètres $\boldsymbol{\rho}$ une partition colonnes. Enfin, on génère la valeur de chaque x_{ij} suivant la distribution $f_{z_i,w_j}(.;\alpha)$. Cette opération est réalisée par la fonction *GaussBlocRnd* qui génère aléatoirement à partir des paramètres $\boldsymbol{\theta}$ une étiquette pour chaque ligne, puis les étiquettes des colonnes (fonction *PartRnd*). Enfin, on génère les valeurs des x_{ij} suivant une gaussienne. On obtient ainsi un tableau de données dont on connaît les vraies partitions et la vraie valeur des paramètres. Cela permet par la suite une comparaison entre les partitions « réelles » et les partitions estimées.

2.2 Modèle de bloc latent pour la classification croisée à données quantitatives continues

On complète le modèle exposé dans la section précédente en considérant que la distribution $f_{kl}(x_{ij}, \boldsymbol{\alpha})$ de chaque x_{ij} suit une loi gaussienne. On obtient ainsi des données quantitatives continues dont la distribution est définie par :

$$\begin{aligned} f_{kl}(x_{ij}; \boldsymbol{\alpha}) &= \varphi(x_{ij}; \mu_{kl}, \sigma^2_{kl}) \\ &= \frac{1}{\sqrt{2\pi\sigma^2_{kl}}} \exp\Big(-\frac{1}{2\sigma^2_{kl}}(x_{ij} - \mu_{kl})^2 \Big) \end{aligned}$$

avec $\boldsymbol{\alpha} = (\mu_{11}, ...\mu_{gm}, \sigma^2_{11}, ..., \sigma^2_{gm})$.

Afin de déterminer le couple de partitions $(\mathbf{z}, \mathbf{w})$, on cherche à maximiser la fonction de vraisemblance du modèle. Cette dernière est intraitable sauf pour des petites valeurs de n et d. Cependant, une approximation de cette fonction a été proposée par Govaert Govaert and Nadif (2009). La vraisemblance classifiante, aussi appelée vraisemblance des données complétée, est définie à la constante près $-\frac{nd}{2} \log 2\pi$ par :

$$L_c(\mathbf{z}, \mathbf{w}, \boldsymbol{\theta}) = \sum_k z_k \log \pi_k + \sum_l w_l \log \rho_l + L_{CR}(\mathbf{z}, \mathbf{w}, \boldsymbol{\alpha})$$

où la vraisemblance classifiante restreinte est :

$$\begin{aligned} L_{CR}(\mathbf{z}, \mathbf{w}, \boldsymbol{\alpha}) &= \sum_{i,j,k,l} z_{ik} w_{jl} \log f(x_{ij}; \boldsymbol{\alpha}_{kl}) \\ &= -\frac{1}{2} \sum_{i,j,k,l} z_{ik} w_{jl} (\log \sigma^2_{kl} + \frac{1}{\sigma^2_{kl}}(x_{ij} - \mu_{kl})^2) \\ &= -\frac{1}{2} \sum_{k,l} z_k w_l \log \sigma^2_{kl} - \frac{1}{2} \sum_{k,l} \frac{1}{\sigma^2_{kl}} \sum_{i,j} z_{i,k} w_{j,l} (x_{ij} - \mu_{kl})^2 \end{aligned}$$

La fonction de vraisemblance classifiante restreinte est utilisée dans l'algorithme EM pour la détermination du couple de partitions $(\mathbf{z}, \mathbf{w})$. (Neal and Hinton, 1998) en proposent une interprétation. Ils présentent l'algorithme EM comme un algorithme itératif en deux temps : on alterne une phase d'estimation avec une phase de maximisation obtenant après itérations le couple de partitions $(\mathbf{z}, \mathbf{w})$. Dans le cas de la classification croisée, on l'étend à trois temps. Dans le premier, on réalise une maximisation de la fonction de vraisemblance classifiante pour $\boldsymbol{\theta}$ et $\mathbf{w}$ fixés. On trouve ainsi un $\mathbf{z}$. Puis, pour $\boldsymbol{\theta}$ et $\mathbf{z}$ fixés, on estime $\mathbf{w}$. Enfin, on maximise $L_{CR}(\mathbf{z}, \mathbf{t}, \boldsymbol{\alpha})$ pour $\mathbf{z}$ et $\mathbf{w}$ fixé. On répète ces 3 opérations jusqu'à convergence de l'algorithme en un maximum.

3 La simulation d'un jeu de données

On s'intéresse maintenant à la simulation d'un jeu de données suivant le modèle défini dans la partie précédente. Pour cela, on considère un tableau de taille $n \times d$, le nombre de classes $g \times m$ ainsi que les paramètres du modèle $\boldsymbol{\theta}$. On génère aléatoirement les classes de chaque lignes z_i et de chaque colonne w_j. A partir de ces partitions, on simule les valeurs du tableau x_{ij}. Par la suite, on mesure la qualité du mélange des données.

3.1 Les classifieurs

Pour mesurer la qualité du mélange des données, on définit deux classifieurs :
- la probabilité a posteriori,
- le classifieur de Bayes muni et une fonction coût L.

3.1.1 Classifieur MAP

On veut affecter à toute observation la classe ayant la plus grande probabilité a posteriori. Cette affectation est considérée optimale, puisqu'elle minimise le nombre d'erreurs d'affectation. L'objectif est de déterminer le couple de partitions $(\mathbf{z}, \mathbf{w})$ maximisant cette probabilité a posteriori notée $f(\mathbf{z}, \mathbf{w}|\mathbf{x}, \boldsymbol{\theta})$. $(\mathbf{z}, \mathbf{w})$ est obtenu en maximisant la log-vraisemblance classifiante.

Le classifieur MAP est défini par :

$$\mathbf{h}_{MAP}(\mathbf{x}) = \arg\max_{\mathbf{z},\mathbf{w}} \mathbb{P}(\mathbf{z}, \mathbf{w}|\mathbf{x})$$

A ce classifieur, on associe une fonction coût afin d'en mesurer sa qualité. Elle calcule la perte engendrée par le choix de $(\mathbf{z'}, \mathbf{w'})$ alors que la vraie partition est $(\mathbf{z}, \mathbf{w})$.

$$\mathbf{L} : (Z \times W) \times (Z \times W) \to \mathbb{R} : \mathbf{L}((\mathbf{z}, \mathbf{w}), (\mathbf{z'}, \mathbf{w'}))$$

De cette fonction coût, on détermine le risque défini par : $R(\mathbf{h}) = \mathbb{E}[\mathbf{L}(\mathbf{h}(\mathbf{x}), (\mathbf{z}, \mathbf{w}))]$

3.1.2 Classifieur de Bayes

Il s'agit d'un classifieur probabiliste basé sur le théorème de Bayes muni de l'hypothèse d'indépendance des classes.

$$\mathbf{h}_{Bayes}(\mathbf{x}) = \arg\min_{\mathbf{h}} \mathbb{E}[\mathbf{L}(\mathbf{h}(\mathbf{x}), (\mathbf{z}, \mathbf{w}))|\mathbf{x}] \forall \mathbf{x}$$

3.1.3 Choix de la fonction coût

On pose $\mathbf{L}_1$, la fonction coût mesurant le degré de mélange du modèle. Il s'agit du pourcentage de mal classés exprimé par :

$$\begin{aligned} \mathbf{L}_1((\mathbf{z}, \mathbf{w}), (\mathbf{z'}, \mathbf{w'})) &= 1 - \frac{1}{nd} \sum_{i,j} \delta_{(z_i, w_j), (z'_i, w'_j)} \\ &= 1 - \frac{1}{nd} \sum_{i,j,k,l} z_{ik} z'_{ik} w_{jl} w'_{jl} \end{aligned}$$

Le classifieur associé de Bayes est alors :

$$\begin{aligned}\mathbf{h}_{Bayes}(\mathbf{x}) &= \arg\min_{\mathbf{h}} \mathbb{E}[\mathbf{L}_1(\mathbf{h}(\mathbf{x}),(\mathbf{z},\mathbf{w}))|\mathbf{x}] \\ &= \arg\min_{\mathbf{h}} 1 - \frac{1}{nd}\sum_{i,j}\mathbb{P}(z_i = h_{1i}(\mathbf{x}), w_j = h_{2j}(\mathbf{x})|\mathbf{x}) \\ &= \arg\max_{\mathbf{z},\mathbf{w}} \sum_{i,j}\mathbb{P}(z_i, w_j|\mathbf{x})\end{aligned}$$

Le risque associé s'écrit alors : $R(\mathbf{h}_{Bayes}) = 1 - \mathbb{E}[\max_{\mathbf{z},\mathbf{w}} \frac{1}{nd}\sum_{i,j}\mathbb{P}(z_i, w_j|\mathbf{x})]$.

Le classifieur de Bayes munis de $\mathbf{L}_1$ semble différent du classifieur MAP. En revanche, en prenant la fonction coût $\mathbf{L}_2 = 1 - \delta_{(z,w),(z',w')}$, les deux classifieurs sont égaux. Néanmoins, $\mathbf{L}_2$ semble peu applicable dans le traitement des données.

Par ailleurs, le classifieur MAP muni de $\mathbf{L}_1$ ne s'écrit pas sous une forme explicite. C'est pourquoi pour la suite de l'étude, on prend le classifieur de Bayes qui minimise le risque associé à $\mathbf{L}_1$ connaissant les données, pour comparer le couple de vraies partitions et celui déduit de l'estimation MAP.

3.2 L'erreur MAP

Avec des données simulées, on peut calculer le degré de mélange du modèle. Pour ce faire, on simule un jeu pour lequel on connaît les vraies partitions. Puis à ce jeu, on calcule le MAP associé à chaque valeur. On détermine ainsi une classe ligne et une classe colonne pour chaque valeur du tableau. On compare les partitions ainsi obtenues avec les vraies. On obtient ainsi un pourcentage de mal classé que l'on nomme « erreur MAP ». Elle est calculée à partir de l'erreur commise en ligne $e(\mathbf{z},\mathbf{z}')$ et l'erreur en colonne $e(\mathbf{w},\mathbf{w}')$:

$$\mathbf{e}((\mathbf{z},\mathbf{w}),(\mathbf{z}',\mathbf{w}')) = e(\mathbf{z},\mathbf{z}') + e(\mathbf{w},\mathbf{w}') - e(\mathbf{z},\mathbf{z}')e(\mathbf{w},\mathbf{w}')$$

avec

$$e(\mathbf{z},\mathbf{z}') = 1 - \frac{1}{n}\sum_i\sum_k z_{ik}z'_{ik} \tag{4}$$

(Govaert and Nadif, 2008) définissent ainsi 3 cas :

- les classes sont séparées, ce qui correspond à un pourcentage de mal classés égal à 5%,
- les classes sont moyennement séparées soit un pourcentage de 12%,
- les classes sont peu séparées soit un pourcentage de l'ordre de 20%.

4 Étude de la variation de l'erreur MAP

4.1 Étude avec n, d croissants et variances égales

4.1.1 Cas symétrique

Dans ce paragraphe, on étudie le cas d'un tableau symétrique où le nombre de lignes est égal à celui des colonnes. Il en est de même avec le nombre de classes en lignes et de classes en colonnes.

Les données, les paramètres

On considère un tableau de taille $n \times d$ composé de $g = 3$ classes lignes et $m = 3$ classes colonnes. n, le nombre de lignes, et d, le nombre de colonnes, varient entre 50 et 1000. On fixe le paramètre μ tel que les moyennes de chaque classe soient équidistantes.

$$\boldsymbol{\mu} = \begin{pmatrix} 1 & -1 & -1 \\ -1 & -1 & 1 \\ -1 & 1 & -1 \end{pmatrix}$$

On a donc un μ fixe et σ^2 qui varie afin d'obtenir un pourcentage de mal classés correspondant aux trois cas. On obtient ainsi $\sigma^2 = (12.5; 16.54; 20.5)$ dont les valeurs correspondent respectivement à 5%, 12% et 20%. Ces valeurs sont obtenues pour des tableaux de taille 100×100. Ces paramètres permettent d'avoir une erreur en ligne du même ordre que l'erreur en colonne entraînant ainsi un « vrai » problème de classification croisée.

Les figures 2 à 7 donnent un exemple de représentation des données pour un pourcentage de mal classés égal à 5%, 12% et 20%. Le MAP est calculé avec la fonction « GaussBlocMap ». Pour chaque taille de n et d, on réalise 200 de simulations. En effet, pour des valeurs identiques des paramètres du modèle, on obtient une variabilité dans le calcul du taux d'erreur autour d'une valeur moyenne. On réalise donc plusieurs simulations ayant un même modèle afin d'obtenir une valeur moyenne égale aux différents taux voulus. Par exemple, pour des classes moyennement séparées, un tableau de tille 100×100 et 200 simulations, on obtient une répartition du taux d'erreur centré autour de 12% (1).

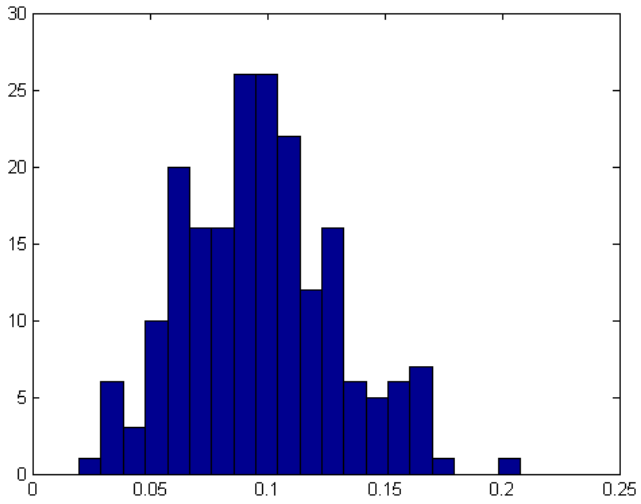

FIG. 1 – Histogramme de taux d'erreur MAP pour 200 simulations et des classes moyennement séparées

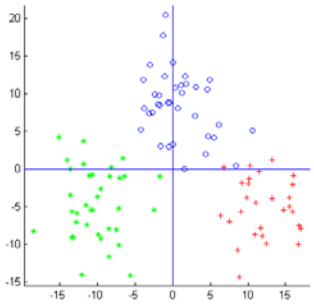

FIG. 2 – ACP des lignes pour un taux de 5% (cas symétrique)

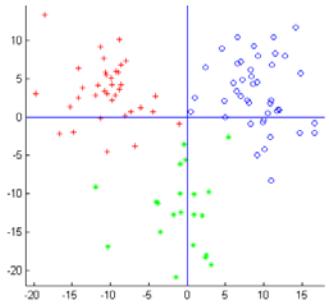

FIG. 3 – ACP des colonnes pour un taux de 5% (cas symétrique)

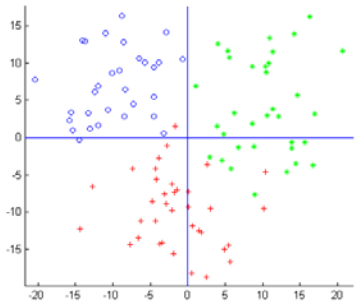

FIG. 4 – ACP des lignes pour un taux de 12% (cas symétrique)

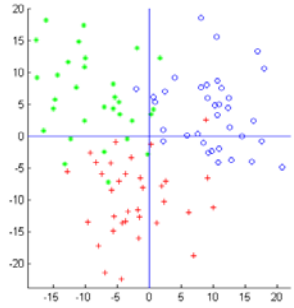

FIG. 5 – ACP des colonnes pour un taux de 12% (cas symétrique)

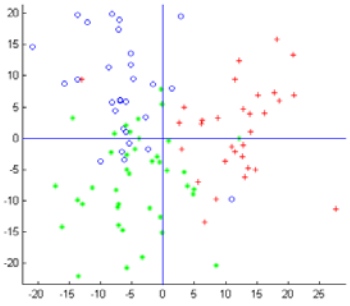

FIG. 6 – ACP des lignes pour un taux de 20% (cas symétrique)

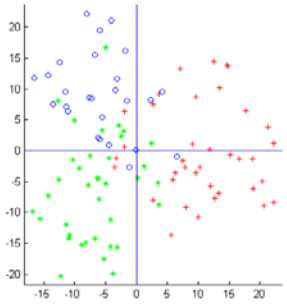

FIG. 7 – ACP des colonnes pour un taux de 20% (cas symétrique)

Étude du taux de mal classés MAP

L'algorithme itératif permettant de calculer l'erreur MAP est ici initialisé soit avec des partitions simulées aléatoirement (200 essais) soit avec les vraies partitions. Un nombre trop petit d'essais, par exemple 10, pour le premier cas donne un calcul d'erreur erroné. Les figures 8, 9 et 10 représentent l'évolution du pourcentage de mal classés quand n et d augmentent suivant les deux types d'initialisation. On remarque que, dans chaque cas, le taux d'erreur de mal classés Map tend vers 0 (au delà de 500, il est égal à 0). De plus, en initialisant l'algorithme avec les « vraies » partitions, le taux d'erreur est moindre qu'en initialisant avec des partitions simulées pour des tableaux de petites tailles (cf tableaux en annexe). Cela s'explique par l'estimation de maximum local.

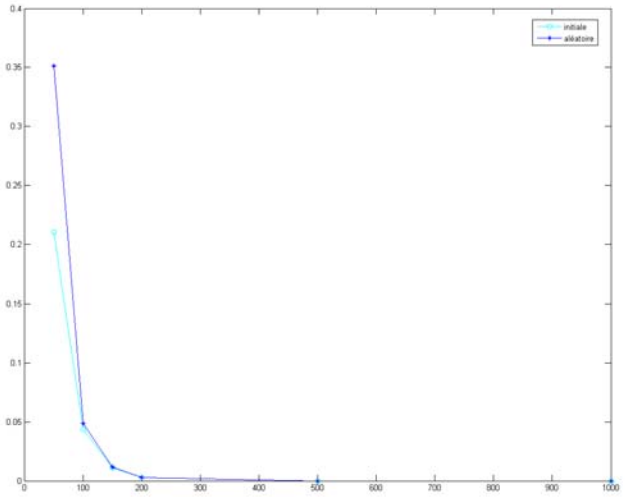

FIG. 8 – Comparaison erreur moyenne « aléatoire » et « initiale » t=0.05 (cas symétrique)

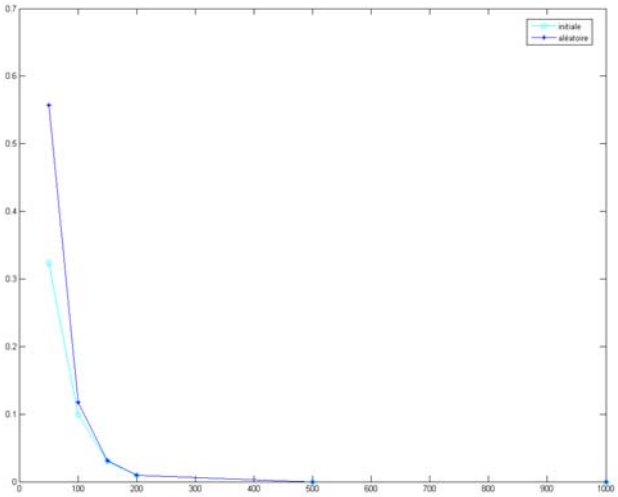

FIG. 9 – Comparaison erreur moyenne « aléatoire » et « initiale » t=0.12 (cas symétrique)

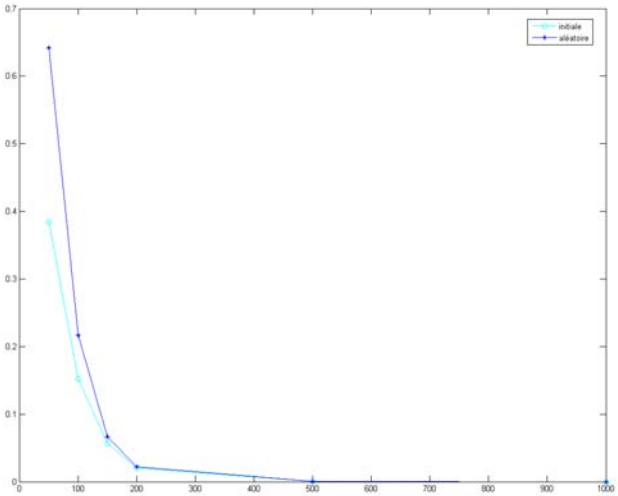

FIG. 10 – Comparaison erreur moyenne « aléatoire » et « initiale » t=0.20 (cas symétrique)

4.1.2 Cas non symétrique

On étudie, dans cette partie, le cas d'un tableau rectangulaire, non symétrique où le nombre de lignes est différent de celui des colonnes. De même, le nombre de classes en lignes est différent de celui des classes en colonnes.

Les données, les paramètres

On considère un tableau de taille $n \times d$ composé de $g = 3$ classes lignes et $m = 2$ classes colonnes. n et d varient respectivement, entre 100-1050 et 50-1000. On fixe le paramètre μ, comme précédemment ; i.e. : les moyennes de chaque classe sont équidistantes.

$$\boldsymbol{\mu} = \begin{pmatrix} 0 & 1 \\ -\sqrt{3}/2 & -1/2 \\ \sqrt{3}/2 & -1/2 \end{pmatrix}$$

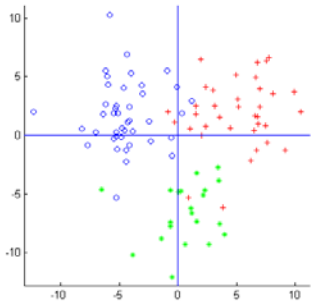

FIG. 11 – ACP des lignes pour un taux de 5% (cas non symétrique)

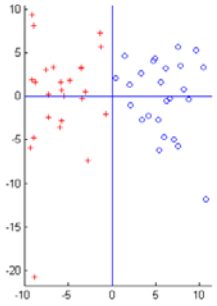

FIG. 12 – ACP des colonnes pour un taux de 5% (cas non symétrique)

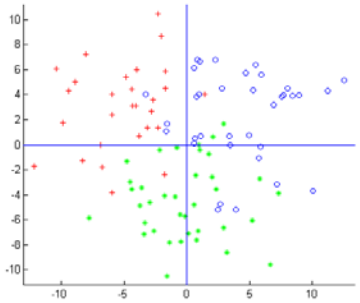

FIG. 13 – ACP des lignes pour un taux de 12% (cas non symétrique)

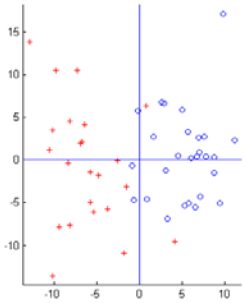

FIG. 14 – ACP des colonnes pour un taux de 12% (cas non symétrique)

Comme pour la section précédente, on s'intéresse à trois valeurs de taux d'erreur 5%, 12% et 20% obtenant ainsi $\sigma^2 = (4.68; 6.8; 9.12)$ pour des tableaux de taille 100×50.

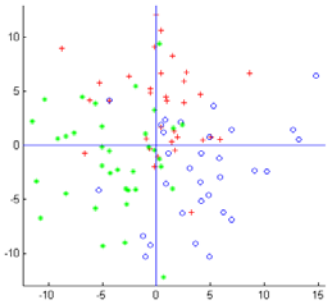

FIG. 15 – ACP des lignes pour un taux de 20% (cas non symétrique)

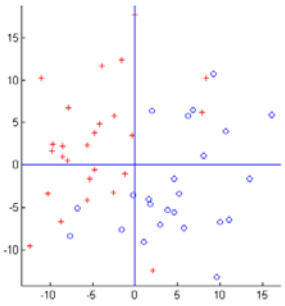

FIG. 16 – ACP des colonnes pour un taux de 20% (cas non symétrique)

Étude du taux de mal classés MAP

La fonction *GaussBlocMap* permettant de calculer le Map est initialisé soit avec des partitions simulées aléatoirement soit avec les vraies partitions. Les figures 17, 18 et 19, représentent l'évolution du pourcentage de mal classés quand n et d augmentent (même pas). On remarque que, dans chaque cas, le taux d'erreur de mal classés Map tend vers 0 (au delà de 500, il est égale à 0). Comme précédemment, en initialisant l'algorithme avec les « vraies » partitions, le taux d'erreur est inférieur à celui qui est calculé en initialisant l'algorithme avec des partitions simulées (cf tableaux annexes).

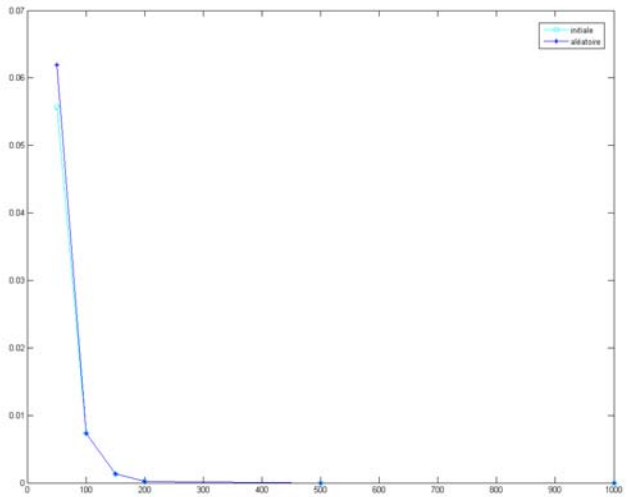

FIG. 17 – Évolution du pourcentage de mal classés quand n et d augmentent pour des classes séparées (cas non symétrique)

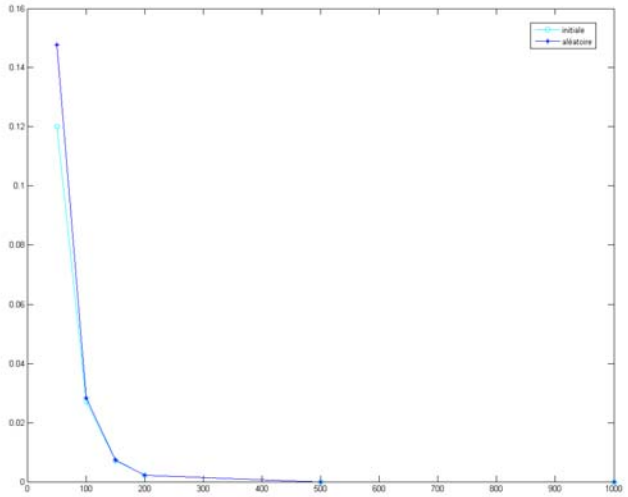

FIG. 18 – Évolution du pourcentage de mal classés quand n et d augmentent pour des classes moyennement séparées (cas non symétrique)

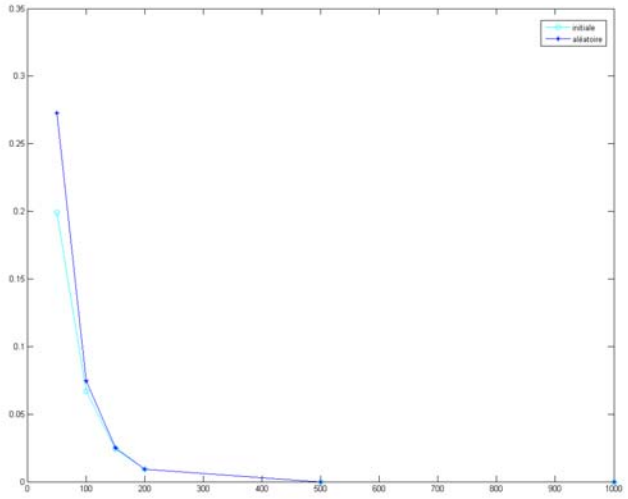

FIG. 19 – Évolution du pourcentage de mal classés quand n et d augmentent pour des classes peu séparées (cas non symétrique)

Conclusion

L'erreur MAP dépend de la taille du tableau et des paramètres. Pour obtenir des classes très, moyennement et peu séparées, on doit donc tenir compte de la taille du jeu de données simulées. De plus, quand n et d augmentent, l'erreur

MAP semble tendre vers 0. Enfin, l'estimation du Map calculée à partir des vraies partitions est meilleure que celle calculée à partir de partition aléatoire. Cependant, pour des tableaux de taille suffisante, les résultats sont les mêmes quelque soit l'initialisation.

4.2 Étude pour n croissant et d fixé

On réalise la même démarche que précédemment (section 4.1.2) mais on ne fait varier qu'une seule dimension du tableau, n. d est fixé à 50. On remarque que l'erreur calculée à partir des vraies partitions est inférieure à celle calculée avec des partitions aléatoires. Par ailleurs, on trouve que l'erreur colonne tend vers 0 et donc l'erreur MAP tend vers l'erreur ligne. Cela s'explique par l'augmentation du nombre d'informations en colonne. On note une légère augmentation de la courbe pour deux des graphes suivants. Une étude plus approfondie (avec des tailles plus grandes de n) révèle qu'il s'agit d'oscillations autour d'une valeur théorique du taux d'erreur en ligne.

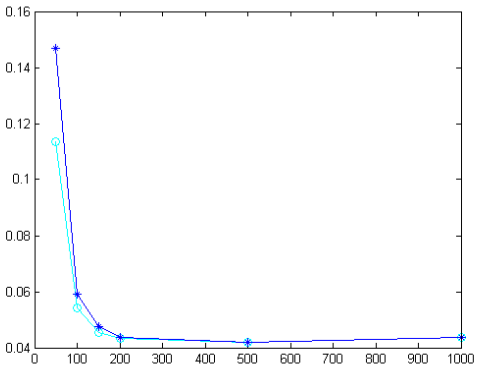

FIG. 20 – Évolution du taux pour des classes séparées

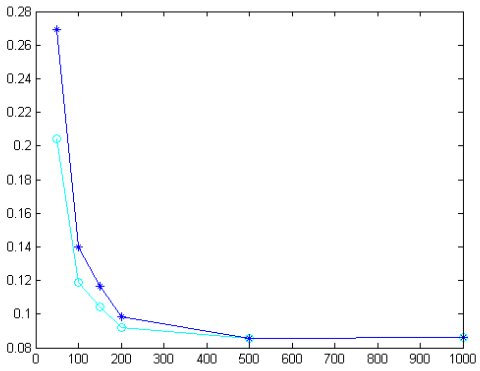

FIG. 21 – Évolution du taux pour des classes moyennement séparées

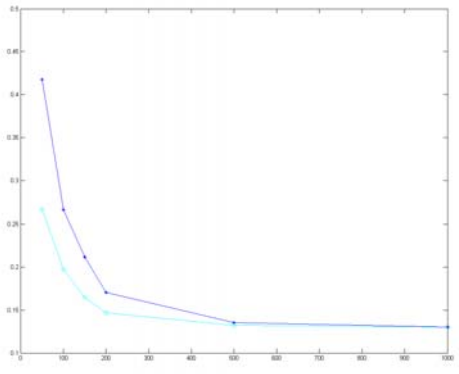

FIG. 22 – Évolution du taux pour des classes peu séparées

4.3 Cas des variances différentes

On réalise la même étude que précédemment mais avec des variances différentes par classes. On choisit des matrices dont les valeurs forment les sommets d'un triangle équilatéral Dans le cas symétrique, on obtient respectivement pour chaque taux d'erreur (5%, 12%, 20%) :

$$\sigma_{05}^2 = \begin{bmatrix} 15,7245 & 12,8655 & 12,8655 \\ 12,8655 & 12,8655 & 15,7245 \\ 12,8655 & 15,7245 & 12,8655 \end{bmatrix} \sigma_{12}^2 = \begin{bmatrix} 22 & 18 & 18 \\ 18 & 18 & 22 \\ 18 & 22 & 18 \end{bmatrix} \sigma_{20}^2 = \begin{bmatrix} 30,1466 & 24,6654 & 30,1466 \\ 24,6654 & 24,6654 & 24,6654 \\ 24,6654 & 30,1466 & 24,6654 \end{bmatrix}$$

Dans le cas non symétrique, on a :

$$\sigma_{05}^2 = \begin{bmatrix} 100 & 165 \\ 43,7083 & 67,5 \\ 156,2916 & 67,5 \end{bmatrix} \sigma_{12}^2 = \begin{bmatrix} 100 & 154 \\ 53,2346 & 73 \\ 146,7654 & 73 \end{bmatrix} \sigma_{20}^2 = \begin{bmatrix} 100 & 146,3 \\ 59,9030 & 76,85 \\ 140,0969 & 76,85 \end{bmatrix}$$

On obtient des résultats similaires à la section 4.1 et 4.2. En effet, que soit dans le cas symétrique ou non symétrique, quand les tailles de n et d augmentent, le taux d'erreurs MAP tend vers 0. Quand n croît et d fixé, on note que l'erreur MAP tend en oscillant vers la valeur de l'erreur ligne théorique. Ce résultat est illustré dans la figure 23. Sur cette figure, on y compare le taux d'erreur MAP moyen calculé en partant des partitions initiales avec celui calculé avec la partition colonne connue et bloquée : seule le MAP de la partition en ligne est estimé. On obtient une erreur colonne fixe (égale à 0) à laquelle on compare l'erreur moyenne.

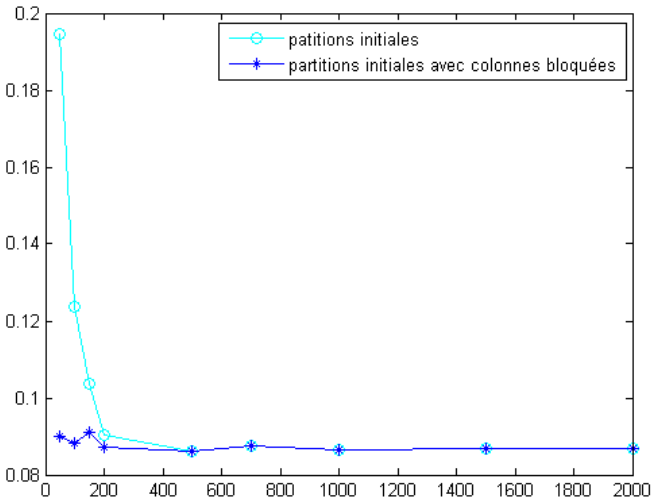

FIG. 23 – Évolution du pourcentage de mal classés quand n augmente, d fixé et dont les partitions initiales sont connues.

Enfin, l'estimation de l'erreur MAP en initialisant l'algorithme avec les vraies valeurs des partitions donne une erreur plus faible que celle avec les partitions

générées aléatoirement pour des tableaux de petite taille. Pour des tableaux 200×200 et quelque soit la qualité de mélange, l'estimation de l'erreur MAP donne une valeur similaire selon le type d'initialisation de l'algorithme.

5 Conclusion

Le modèle de « block clustering » est une méthode de classification croisée de données continues dont les partitions recherchées sont des variables latentes. L'objectif est de définir un couple de partitions lignes-colonnes permettant après réorganisation du tableau suivant ces partitions d'obtenir des blocs homogènes.

En perspective d'un plan expérience correspondant aux hypothèses de travail (afin d'étudier des critères de sélection de modèles), on réalise un protocole de simulation. L'illustration et la validation de résultats par des jeux de données pertinents est une étape nécessaire lors de la réalisation de travaux en recherche. Il est donc important de contrôler les simulations des jeux de données issues du modèle de mélange pour la classification croisée. Néanmoins, cette étude amène à un vrai problème de calcul de l'erreur MAP. En effet, l'erreur de Bayes munie de la fonction coût L_1 ne semble pas toujours égale à l'erreur MAP. On en propose alors une approximation par le pourcentage de mal classés. Ce critère semble efficace et donne un bon indicateur du degré du mélange. Cela permet d'avoir des problèmes de classification dont on peut évaluer la complexité.

Une fois l'erreur MAP définie, on réalise un protocole de simulation de jeux de données. On considère trois catégories de mélange de données : les classes sont soit séparées ce qui correspond à une erreur MAP égale à 5%, soit moyennement séparées (12%) ou encore peu séparées (20%). Pour définir ces catégories, on fixe les paramètres du modèle de blocs latents ainsi que la taille du tableau à l'exception de la variance. En faisant varier la variance du modèle, on a différentes qualités de mélange des données.

L'intérêt des simulations est multiple. Parce qu'on a simulé un jeu de données complet, on a la vraie valeur des partitions. On peut ainsi évaluer les performances de l'algorithme utilisé en calculant la distance entre les partitions initiales et les partitions obtenues après estimation. De plus, on valide ainsi le modèle de classification : on part d'un modèle, utilisé pour la simulation ; puis on estime les paramètres du modèle en utilisant le jeu de données ; enfin on vérifie que le modèle obtenu après cette opération est semblable à celui de départ. Enfin, le contrôle des simulations amène à un « vrai » problème de classification croisée : on crée ainsi un problème de classification en ligne et en colonne de même complexité.

Pour finir, il sera intéressant pour l'utilisation d'un jeu de données réelles qu'il soit de taille suffisamment grande afin d'éviter le problème de surestimation du degrés de mélange due à l'initialisation de l'algorithme avec des partitions aléatoires.

6 Annexes

6.1 Étude n et d croissants et variances égales

6.1.1 Cas symétrique

$t = 0.05$

n, d	50	100	150	200	500	1000
err alea moy	0.3513	0.0488	0.0114	0.0029	0	0
err alea row	0.1992	0.0251	0.0054	0.0019	0	0
err alea col	0.2041	0.0243	0.0061	0.001	0	0
err alea stdmoy	0.1619	0.0275	0.0097	0.0042	0	0
err init moy	0.2102	0.0435	0.0112	0.0027	0	0
err init row	0.1114	0.0223	0.0052	0.0018	0	0
err init col	0.1127	0.0217	0.006	0.0009	0	0
err init stdmoy	0.0793	0.0243	0.0095	0.0038	0	0

$t = 0.12$

n, d	50	100	150	200	500	1000
err alea moy	0.5566	0.1175	0.0315	0.0099	0	0
err alea row	0.345	0.0607	0.0149	0.0048	0	0
err alea col	0.3396	0.061	0.0169	0.0051	0	0
err alea stdmoy	0.1411	0.0473	0.0139	0.0083	0	0
err init moy	0.3237	0.0986	0.0305	0.0097	0	0
err init row	0.1791	0.0509	0.0144	0.0047	0	0
err init col	0.1781	0.0504	0.0163	0.005	0	0
err init stdmoy	0.0851	0.036	0.014	0.0078	0	0

$t = 0.20$

n, d	50	100	150	200	500	1000
err alea moy	0.6419	0.2169	0.0661	0.022	0.0003	0
err alea row	0.4079	0.118	0.0337	0.0104	0.0002	0
err alea col	0.4116	0.1146	0.0336	0.0117	0.0002	0
err alea stdmoy	0.1253	0.0848	0.0275	0.0086	0.0007	0
err init moy	0.3841	0.1522	0.0563	0.0205	0.0003	0
err init row	0.2139	0.0826	0.0285	0.0097	0.0002	0
err init col	0.2194	0.0763	0.0287	0.0109	0.0002	0
err init stdmoy	0.0895	0.0464	0.023	0.0089	0.0007	0

6.1.2 Cas non symétrique

$t = 0.05$

n	100	150	200	250	550	1050
d	50	100	150	200	500	1000
err alea moy	0.0619	0.0074	0.0013	0.0002	0	0
err alea row	0.0468	0.005	0.0006	0	0	0
err alea col	0.0161	0.0024	0.0007	0.0002	0	0
err alea stdmoy	0.0352	0.0077	0.0027	0.001	0	0
err init moy	0.0557	0.0074	0.0013	0.0002	0	0
err init row	0.0442	0.005	0.0006	0	0	0
err init col	0.0121	0.0024	0.0007	0.0002	0	0
err init stdmoy	0.0304	0.0077	0.0027	0.001	0	0

$t = 0.12$

n	100	150	200	250	550	1050
d	50	100	150	200	500	1000
err alea moy	0.149	0.0325	0.0055	0.0023	0	0
err alea row	0.1021	0.0201	0.0032	0.0011	0	0
err alea col	0.0535	0.0127	0.0023	0.0012	0	0
err alea stdmoy	0.0645	0.0181	0.0055	0.0035	0	0
err init moy	0.123	0.0305	0.0055	0.0021	0	0
err init row	0.0922	0.0192	0.0032	0.001	0	0
err init col	0.034	0.0115	0.0023	0.0011	0	0
err init stdmoy	0.0396	0.0167	0.0055	0.0034	0	0

$t = 0.20$

n	100	150	200	250	550	1050
d	50	100	150	200	500	1000
err alea moy	0.2725	0.0747	0.0249	0.009	0	0
err alea row	0.174	0.0451	0.0141	0.0042	0	0
err alea col	0.1237	0.0311	0.011	0.0049	0	0
err alea stdmoy	0.1039	0.0279	0.0138	0.006	0	0
err init moy	0.1989	0.0673	0.0241	0.009	0	0
err init row	0.1405	0.0418	0.0137	0.0042	0	0
err init col	0.0683	0.0266	0.0105	0.0049	0	0
err init stdmoy	0.0542	0.023	0.0129	0.006	0	0

6.2 Etude pour n croissant, $d = 50$ et variance égale

$t = 0.05$

n	50	100	150	200	500	1000
err alea moy	0.147	0.059	0.0476	0.0439	0.0419	0.0435
err alea row	0.068	0.046	0.0433	0.0429	0.0419	0.0435
err alea col	0.0868	0.0138	0.0046	0.001	0	0
err alea stdmoy	0.0854	0.0306	0.021	0.0163	0.0102	0.0074
err init moy	0.1134	0.0542	0.0453	0.0433	0.0419	0.0435
err init row	0.0518	0.0446	0.0427	0.0429	0.0419	0.0435
err init col	0.0654	0.0102	0.0028	0.0004	0	0
err init stdmoy	0.0548	0.0277	0.0195	0.0157	0.0102	0.0074

$t = 0.12$

n	50	100	150	200	500	1000
err alea moy	0.2689	0.1401	0.1166	0.0984	0.0854	0.0863
err alea row	0.1386	0.0956	0.0965	0.0909	0.0854	0.0863
err alea col	0.157	0.05	0.0224	0.0084	0	0
err alea stdmoy	0.1208	0.0586	0.0371	0.0256	0.0115	0.0092
err init moy	0.2045	0.1188	0.1042	0.092	0.0854	0.0863
err init row	0.1032	0.0862	0.0927	0.0893	0.0854	0.0863
err init col	0.114	0.0358	0.0128	0.003	0	0
err init stdmoy	0.0713	0.0426	0.0319	0.0209	0.0115	0.0092

$t = 0.20$

n	50	100	150	200	500	1000
err alea moy	0.4178	0.2669	0.212	0.1706	0.1352	0.13
err alea row	0.2342	0.1707	0.158	0.1413	0.1331	0.13
err alea col	0.251	0.1204	0.0662	0.0346	0.0024	0
err alea stdmoy	0.1483	0.1045	0.0766	0.0431	0.0165	0.0101
err init moy	0.2675	0.1971	0.1648	0.1465	0.1325	0.13
err init row	0.1466	0.1419	0.1398	0.1351	0.1325	0.13
err init col	0.143	0.0652	0.0294	0.0132	0	0
err init stdmoy	0.0837	0.061	0.0448	0.0284	0.014	0.0101

6.3 Etude n et d croissants et variances différentes

6.3.1 Cas symétrique

$t = 0.05$

n	50	100	150	200	500	1000
err alea moy	0.3825	0.0605	0.014	0.0026	0	0
err alea row	0.2199	0.0315	0.0065	0.001	0	0
err alea col	0.2226	0.0302	0.0075	0.0016	0	0
err alea stdmoy	0.1611	0.0317	0.009	0.0038	0	0
err init moy	0.2248	0.0509	0.0139	0.0026	0	0
err init row	0.1201	0.0263	0.0065	0.001	0	0
err init col	0.1211	0.0253	0.0075	0.0016	0	0
err init stdmoy	0.0862	0.0258	0.0091	0.0038	0	0

$t = 0.12$

n	50	100	150	200	500	1000
err alea moy	0.5934	0.1453	0.0447	0.0136	0	0
err alea row	0.3654	0.0755	0.0229	0.0068	0	0
err alea col	0.3774	0.0764	0.0223	0.0068	0	0
err alea stdmoy	0.1423	0.0582	0.0205	0.0085	0	0
err init moy	0.3391	0.1145	0.0427	0.0131	0	0
err init row	0.1858	0.0606	0.0216	0.0066	0	0
err init col	0.1905	0.0576	0.0216	0.0065	0	0
err init stdmoy	0.0853	0.037	0.02	0.0078	0	0

$t = 0.20$

n	50	100	150	200	500	1000
err alea moy	0.6738	0.3178	0.0952	0.0411	0.001	0
err alea row	0.4359	0.1763	0.0495	0.0197	0.0008	0
err alea col	0.4352	0.1771	0.0482	0.0219	0.0002	0
err alea stdmoy	0.1088	0.1089	0.0325	0.0183	0.0011	0
err init moy	0.4082	0.2044	0.0836	0.039	0.001	0
err init row	0.2337	0.1085	0.0435	0.0188	0.0008	0
err init col	0.2311	0.1085	0.0421	0.0206	0.0002	0
err init stdmoy	0.0925	0.0604	0.0281	0.0177	0.0011	0

6.3.2 Cas non symétrique

$t = 0.05$

n	50	100	150	200	500	1000
err alea moy	0.0659	0.0097	0.0017	0.0001	0	0
err alea row	0.0449	0.0049	0.001	0	0	0
err alea col	0.0222	0.0048	0.0007	0.0001	0	0
err alea stdmoy	0.0379	0.01	0.0031	0.0007	0	0
err init moy	0.0574	0.009	0.0017	0.0001	0	0
err init row	0.042	0.0046	0.001	0	0	0
err init col	0.0162	0.0045	0.0007	0.0001	0	0
err init stdmoy	0.0293	0.0095	0.0031	0.0007	0	0

$t = 0.12$

n	50	100	150	200	500	1000
err alea moy	0.1657	0.032	0.0091	0.0031	0	0
err alea row	0.1043	0.018	0.0048	0.0011	0	0
err alea col	0.0701	0.0143	0.0043	0.002	0	0
err alea stdmoy	0.0762	0.0208	0.008	0.0039	0	0
err init moy	0.1217	0.0291	0.0086	0.0031	0	0
err init row	0.0861	0.017	0.0046	0.0011	0	0
err init col	0.0392	0.0123	0.0041	0.002	0	0
err init stdmoy	0.0449	0.0185	0.0079	0.0039	0	0

$t = 0.20$

n	50	100	150	200	500	1000
err alea moy	0.2916	0.0787	0.0299	0.0101	0.0002	0
err alea row	0.1807	0.0435	0.0147	0.0045	0	0
err alea col	0.1409	0.037	0.0154	0.0057	0.0002	0
err alea stdmoy	0.1154	0.0312	0.0134	0.0077	0.0006	0
err init moy	0.1939	0.065	0.0276	0.0098	0.0002	0
err init row	0.1377	0.038	0.0137	0.0043	0	0
err init col	0.0655	0.0282	0.014	0.0055	0.0002	0
err init stdmoy	0.0514	0.0248	0.0117	0.0077	0.0006	0

6.3.3 Étude avec n croissant et $d = 50$

$t = 0.05$

n	50	100	150	200	500	1000	1500
err alea moy	0.0399	0.031	0.0281	0.0258	0.0275	0.0255	0.0275
err alea row	0.0286	0.0279	0.0277	0.0254	0.0275	0.0255	0.0275
err alea col	0.0117	0.0032	0.0004	0.0004	0	0	0
err alea stdmoy	0.0257	0.0176	0.0126	0.0104	0.0066	0.0048	0.0049
err init moy	0.0368	0.0299	0.0279	0.0252	0.0275	0.0255	0.0275
err init row	0.0274	0.0277	0.0277	0.0252	0.0275	0.0255	0.0275
err init col	0.0096	0.0023	0.0002	0	0	0	0
err init stdmoy	0.0223	0.016	0.0124	0.0106	0.0066	0.0048	0.0049

$t = 0.12$

n	50	100	150	200	500	1000	1500
err alea moy	0.1605	0.1143	0.0992	0.0926	0.0891	0.079	0.0847
err alea row	0.1014	0.0881	0.0865	0.085	0.0891	0.079	0.0847
err alea col	0.0667	0.0293	0.0142	0.0084	0	0	0
err alea stdmoy	0.0624	0.0505	0.0363	0.022	0.0084	0.0052	0.008
err init moy	0.1175	0.0919	0.085	0.0853	0.0891	0.079	0.0847
err init row	0.0839	0.0805	0.0817	0.0827	0.0891	0.079	0.0847
err init col	0.0369	0.0125	0.0036	0.0028	0	0	0
err init stdmoy	0.043	0.0305	0.0233	0.0174	0.0084	0.0052	0.008

$t = 0.20$

n	50	100	150	200	500	1000	1500
err alea moy	0.3053	0.2432	0.2002	0.1808	0.1469	0.1893	
err alea row	0.1835	0.1656	0.1502	0.1474	0.1351	0.1509	
err alea col	0.1545	0.0955	0.0608	0.04	0.014	0.052	
err alea stdmoy	0.1134	0.0834	0.0749	0.0543	0.0398	0.1476	
err init moy	0.1938	0.158	0.1425	0.1359	0.1305	0.1246	
err init row	0.1337	0.1321	0.1295	0.1307	0.1305	0.1246	
err init col	0.0698	0.03	0.015	0.006	0	0	
err init stdmoy	0.0549	0.0379	0.0303	0.0249	0.0132	0.0071	

Références

Berkhin, P. (2006). A survey of clustering data mining techniques. *Grouping Multidimensional Data*, pages 25–71.

Govaert, G. (1995). Simultaneous clustering of rows and columns. *Control and Cybernetics*, 24(4) :437–458.

Govaert, G. and Nadif, M. (2006). Fuzzy clustering to estimate the parameters of block mixture models. *Soft Computing*, 10(5) :415–422.

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

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

Govaert, G. and Nadif, M. (2009). Un modèle de mélange pour la classification croisée d'un tableau de données continues. In *CAp 09, 11e conférence sur l'apprentissage artificiel*, pages 287–302, Hammamet, Tunisie.

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

Manjunatha, J., Chris, P., Erik, L., Thomas, Z., and David, K. (2007). Analyzing in situ gene expression in the mouse brain with image registration, feature extraction and block clustering. *BMC Bioinformatics*, 8.

Miklós, I., Somodi, I., and Podani, J. (2005). Rearrangement of ecological data matrices via markov chain monte carlo simulation. *Ecology*, 86(12) :3398–3410.

Neal, R. and Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 52 :537–549.

Shan, H. and Banerjee, A. (2008). Bayesian co-clustering. In *Eighth IEEE International Conference on Data Mining, 2008. ICDM'08*, pages 530–539.