REVUE DE STATISTIQUE APPLIQUÉE

Y. BENCHEIKH

Classification croisée et distance L_1 adaptative

Revue de statistique appliquée, tome 50, nº 3 (2002), p. 53-72
<http://www.numdam.org/item?id=RSA_2002__50_3_53_0>

© Société française de statistique, 2002, tous droits réservés.

L'accès aux archives de la revue « Revue de statistique appliquée » (http://www.sfds.asso.fr/publicat/rsa.htm) implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM
Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

Rev. Statistique Appliquée, 2002, **L** (3), 53-72

CLASSIFICATION CROISÉE ET DISTANCE L_1 ADAPTATIVE

Y. BENCHEIKH*

* *Département de mathématiques, Faculté des sciences, Université Ferhat Abbas de Sétif, Sétif 19000 - ALGERIE.*

RÉSUMÉ

Les liens existant entre les méthodes de classification automatique et les modèles de statistique inférentielle ont surtout été étudiés dans le cas de la classification simple.

Nous nous proposons ici de le faire avec une méthode de classification croisée de données binaires. Nous montrons comment l'identification d'un mélange de distributions de Bernoulli avec le même paramètre pour toutes les classes correspond à un critère de classification croisée de données binaires utilisant la distance L_1 et des noyaux binaires. Nous avons généralisé ce modèle en prenant des paramètres qui dépendent des classes en lignes, puis des paramètres qui dépendent des classes en colonnes. Enfin, nous terminons par le cas le plus général : cette fois-ci, les paramètres peuvent varier suivant les classes en lignes et en colonnes : nous proposons alors de nouveaux algorithmes de classification croisée utilisant des distances adaptatives de type L_1.

Mots-clés : *Distance L_1, classification automatique, mélange, classification croisée.*

ABSTRACT

The relation betwen clustering methods and statistical models have been studied in the case of simple clustering methods.

We propose to do it in the case of binary cross clustering method. We show how the identification of mixture of Bernoulli distributions with the same parameter for all classes correspond to a clustering criterion which uses L_1 distance and binary kernels. We have generalized this model, at first by selecting parameters which can depend on line classes, then on column classes and finally by selecting parameters which can depend on line and column classes. Then we propose new cross clustering algorithms using adaptive distances.

Keywords : *L_1 distance, clustering, mixture, cross mixture.*

Introduction

De nombreuses méthodes de classification reposent essentiellement sur la définition d'une métrique et d'un critère associé sans faire référence explicitement à des modèles probabilistes.

En réalité, comme Scott et Symons [12], Schroeder [11] et Celeux [4] le proposent, il est souvent possible de montrer qu'il y a un modèle sous-jacent :

Celui-ci permet alors de donner une interprétation du critère et de justifier son choix. Par exemple Celeux [4] a donné une signification au critère d'inertie interclasse utilisé pour la classification d'individus décrits par des variables quantitatives; Nadif - Marchetti [10] proposent des méthodes de classification pour données qualitatives nominales, liées à un modèle précis de mélange de distributions de probabilité. De même Govaert [8] montre que le critère optimisé par la méthode MNDBIN pour les données binaires correspond à un mélange de lois de Bernoulli.

Toutes ces approches ont étés faite uniquement dans le cas de la classification simple (données qui mettent en jeu un seul ensemble).

Nous proposons dans ce papier de le faire dans le cas de la classification croisée avec une méthode particulière : La méthode CROBIN (classification croisée sur données binaires). Ce travail utilise le lien qui a été établi entre la classification croisée et les modèles probabilistes dans le cas général (Bencheikh [2]); cette dernière montre comment la classification croisée peut être vue comme une solution à un problème d'estimation de paramètres d'un modèle de mélanges et elle développe une méthode de reconnaissance des composants d'un mélange «croisé» que nous étudierons en détail au paragraphe 2.

Dans tout cet article, l'ensemble **I** et l'ensemble **J** décrivant le tableau de données binaires seront considérés et traités de manière identique.

Dans le premier paragraphe, nous décrivons la méthode CROBIN (Govaert [7]) qui est une méthode de classification croisée de données binaires. Dans le second et le troisième paragraphe, nous montrons comment la méthode précédente peut être interprétée comme l'approche classification associée à un mélange de distributions de Bernoulli avec le même paramètre pour toutes les classes; ce modèle correspond au critère de classification croisée binaire utilisant la distance L_1 et des noyaux binaires. Nous étudions dans le paragraphe quatre une extension de ce modèle au cas où le paramètre varie suivant les classes en lignes, les classes en colonnes, puis suivant les classes en lignes et en colonnes; c'est le cas le plus général. En outre, en nous appuyant sur des variantes de ce modèle, nous proposons de nouveaux algorithmes de classification croisée utilisant des distances adaptatives. Enfin nous terminons ce papier par quelques applications pratiques.

1. La méthode CROBIN

1.1. Notations

On notera n et p les cardinaux des deux ensembles **I** et **J**. Le tableau binaire sera noté (x_i^j).

On a donc $x_i^j \in \{0,1\}$. Dans toute la suite les partitions P et Q seront des partitions des deux ensembles **I** et **J** en K et M classes; $P = \{P_1, \ldots, P_K\}$ et $Q = \{Q^1, \ldots, Q^M\}$. K et M sont des constantes fixées arbitrairement.

Soit L l'ensemble des noyaux ou ensemble de représentation. L est l'ensemble des tableaux binaires à K lignes et M colonnes. $L = \{(\lambda_k^m), k = 1, \ldots, K$ et $m = 1, \ldots, M\}$ et $\lambda_k^m \in \{0,1\}$.

λ_k^m représente pour chaque classe la valeur majoritaire.

On note $P \times Q = \{P_k \times Q^m, k = 1, \ldots, K$ et $m = 1, \ldots, M\}$.

1.2. Le problème

La méthode CROBIN (Govaert [7]) fournit une solution locale au problème d'optimisation suivant :

Trouver une partition P de $\boldsymbol{I}$ en K classes, une partition Q de $\boldsymbol{J}$ en M classes et un tableau binaire à K lignes et M colonnes, tel que le critère :

$$\boldsymbol{W}(P \times Q, L) = \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i \in P_k} \sum_{j \in Q^m} |x_i^j - \lambda_k^m| \tag{1}$$

soit minimum.

1.3. L'algorithme

L'algorithme utilisé dans la méthode CROBIN est le suivant : Partant d'un élément (P, Q, L) initial, on fixe Q et on cherche à améliorer P et L, puis on fixe P et on cherche à améliorer Q et L; on construit ainsi une suite (P^n, Q^n, L^n) qui fait décroître le critère $\boldsymbol{W}$. L'algorithme est donc construit à partir de deux étapes intermédiaires que nous allons préciser.

1.3.1. Etapes intermédiaires

Soit P et Q un couple de partitions et L un noyau. Fixons Q et cherchons à améliorer la partition P et le noyau L, c'est-à-dire cherchons une partition P' et un noyau L' tels que : $\boldsymbol{W}(P \times Q, L) > \boldsymbol{W}(P' \times Q, L')$

On peut écrire :

$$\boldsymbol{W}(P \times Q, L) = \sum_{k=1}^{K} \sum_{i \in P_k} \left(\sum_{m=1}^{M} \sum_{j \in Q^m} |x_i^j - \lambda_k^m| \right)$$

Posons $x_i^m = \sum_{j \in Q^m} x_i^j$ et $q_m = \mathrm{Card}(Q^m)$ et notons B la quantité $\sum_{j \in Q^m} |x_i^j - \lambda_k^m|$:

$$B = x_i^m \quad \text{si} \quad \lambda_k^m = 0 \qquad \text{et} \qquad B = q_m - x_i^m \quad \text{si} \quad \lambda_k^m = 1$$

On peut donc écrire $B = |x_i^m - q_m \lambda_k^m|$. D'où l'expression du critère :

$$\boldsymbol{W}(P \times Q, L) = \sum_{k=1}^{K} \sum_{i \in P_k} \sum_{m=1}^{M} |x_i^m - q_m \lambda_k^m|$$

Considérons l'algorithme des Nuées Dynamiques Diday [5] suivant :

– Le tableau de données est le tableau (I, Q) défini par les x_i^m. Les individus sont en lignes et les variables en colonnes. On a donc un ensemble de n éléments et M variables.

– Chacune des quantités $q_m \lambda_k^m$ (avec $1 \leqslant m \leqslant M$ et $\lambda_k^m \in \{0, 1\}$) ne peut être que le minimum ou le maximum atteint par le regroupement des colonnes du tableau initial, c'est à dire les valeurs 0 ou q_m puisque les valeurs initiales étaient 0 ou 1 et il n'y a que q_m colonnes réunies pour former la classe Q^m.

On notera L la matrice des composantes λ_k^m qui définissent l'ensemble des noyaux précédents.

– La distance est la distance «City-block» ou distance L_1.

Le critère habituel de la méthode des Nuées Dynamiques défini à partir des éléments que l'on vient de donner est alors la fonction $\boldsymbol{W}(P \times Q, L)$ que l'on cherche à optimiser. Il suffit de préciser ce que deviennent les fonctions f d'affectation et g de représentation.

Fonction d'affectation (recherche des classes) :

La fonction f d'affectation est toujours la même, chaque élément i est affecté à la classe du noyau de laquelle il est le plus près : i est donc affecté à la classe k minimisant $\sum_{m=1}^{M} |x_i^m - q_m \lambda_k^m|$.

Fonction de représentation (recherche des noyaux) :

On cherche pour toutes les classes P_k, les éléments $(q_1 \lambda_k^1, q_2 \lambda_k^2, \ldots, q_M \lambda_k^M)$ minimisant :

$$\sum_{i \in P_k} \sum_{m=1}^{M} |x_i^m - q_m \lambda_k^m|.$$

On obtient pour m fixé :

$$\sum_{i \in P_k} x_i^m \quad \text{si } \lambda_k^m = 0$$

et

$$\sum_{i \in P_k} (q_m - x_i^m) = n_k q_m - \sum_{i \in P_k} x_i^m \qquad \text{si } \lambda_k^m = 1 \text{ avec } n_k = \mathrm{Card}(P_k).$$

Posons $B' = \sum_{i \in P_k} x_i^m$. Il suffit de prendre comme $m^{\text{ème}}$ composantes du noyau k, la valeur 0 si $B' < n_k q_m - B'$, soit $B' < \frac{n_k q_m}{2}$, sinon on prendra la valeur 1.

Remarquons que cette fonction correspond aussi à la recherche du noyau λ_k^m associé à un couple de classe $P_k \times Q^m$. En effet, les éléments λ_k^m définis suivant

la régle précédente seront 0 ou 1 selon que la somme des éléments intervenant dans la classe $P_k \times Q^m$ sera proche de 0 ou proche du maximum, c'est-à-dire $n_k q_m$. On retrouve ainsi la règle majoritaire. On notera $L'(P, Q)$ le noyau ainsi obtenu.

1.3.2 . Convergence de l'algorithme

L'algorithme CROBIN est le suivant :

A partir d'un triplet (P^0, Q^0, L^0) on applique alternativement les deux étapes intermédiaires; on définit ainsi une suite (P^n, Q^n, L^n) qui vérifie :

$$\boldsymbol{W}(P^0 \times Q^0, L^0) \geqslant \boldsymbol{W}(P^1 \times Q^1, L^1) \geqslant \ldots \geqslant \boldsymbol{W}(P^n \times Q^n, L^n) \geqslant \ldots$$

On peut établir à partir de là les propriétés de convergence habituelles. La suite (P^n, Q^n, L^n) est stationnaire et la valeur du critère associé $\boldsymbol{W}(P^n \times Q^n, L^n)$ décroît strictement jusqu'à la stationnarité. On obtient ainsi une solution localement optimale qui dépend de l'élément de départ choisi. De plus on a montré qu'à chaque étape le noyau L^n est formé en associant à chaque couple de partitions la valeur binaire majoritaire du bloc de 0 et 1 défini par chaque couple de classe. On utilise cette propriété pour simplifier la détermination de l'élément $(P^n \times Q^n, L^n)$: il suffit en effet d'avoir un couple (P^0, Q^0), le noyau L^0 est alors construit à partir de ces partitions.

Pour illustrer le principe de la méthode CROBIN, nous proposons de l'appliquer sur l'exemple suivant.

1.4. Exemple

Soit un ensemble de 10 micro-ordinateurs identifiés par les lettres a à j caractérisés par un ensemble de 10 propriétés identifiées par les nombres 0 à 9. On représente les données initiales (figure 1) sous forme d'un tableau binaire où 1 indique que la propriété est vérifiée et un 0 qu'elle ne l'est pas.

Si on applique l'algorithme CROBIN en demandant trois classes en lignes et deux classes en colonnes, on obtient comme partition de l'ensemble des micro-ordinateurs l'ensemble des classes :

$$P = \Big\{\{a, d, h\}, \{b, e, f, j\}, \{c, g, i\}\Big\}$$

et comme partition de l'ensemble des propriétés l'ensemble des classes :

$$Q = \Big\{\{0, 2, 4, 7, 9\}, \{1, 3, 5, 6, 8\}\Big\}.$$

On désigne par A, B, C les 3 classes de P et par 1, 2 celles de Q.

On peut représenter les partitions sur les données initiales (figure 2) en réordonnant simplement les lignes et les colonnes de manière à respecter les partitions. Les noyaux obtenus sont indiqués sur la figure 3; la valeur du critère est égale à 18 ce

qui indique que sur 100 valeurs initiales du tableau, 18 ne sont pas égales à la valeur idéale représentée par le noyau correspondant.

On peut voir que les micro-ordinateurs de la classe A possèdent en général les propriétés de la classe 1 mais pas celles de la classe 2; ceux de la classe B les propriétés de la classe 2 mais pas celles de la classe 1 et ceux de la classe C aucune propriété. On a résumé le tableau de 100 valeurs par un tableau de 6 valeurs. Les valeurs du tableau des écarts (figure 4) représentent le nombre de fois où la valeur majoritaire n'a pas été prise pour chaque classe, la somme de ces valeurs représente la valeur du critère.

	0123456789
a	1010100101
b	0101011010
c	1000000110
d	1010000100
e	0101111010
f	0100111010
g	0100000101
h	1010110111
i	1001000001
j	0101001000

Fig. 1. – Tableau initial

	02479	13568
a	11111	00000
d	11010	00000
h	11111	00101
b	00000	11111
e	00100	11111
f	00100	10111
j	00000	11010
c	10010	00001
g	00011	10000
i	10001	01000

Fig. 2. – Tableau réordonné

	1	2
A	1	0
B	0	1
C	0	0

Fig. 3. – Tableau des valeurs idéales

	1	2
A	2	2
B	2	3
C	6	3

Fig. 4. – Tableau des écarts

2. Méthode de reconnaissance des composants d'un mélange croisé

La méthode de reconnaissance des composants d'un mélange croisé à été proposée par Bencheikh [1] en 1992; elle s'est intéressée aux méthodes de classification croisée proposées par Govaert [7] et [9]; elle montre que ces méthodes, comme les méthodes de classification simple, peuvent souvent être considérées comme une approche classification d'un modèle de mélange. Pour ceci, elle propose la notion de mélange croisé en se basant sur un exemple concret (Bencheikh [2]) et définit les notions de vraisemblance et de vraisemblance classifiante associées. Elle étudie au passage les liens avec les modèles de mélange simple proposés par Schroeder [11], Celeux [4] et Govaert [8] et montre que ces liens sont tout à fait analogues à ceux qui existent entre les méthodes de classification simple et les méthodes de classification croisée.

Avant de rappeler le principe général de cette méthode, notons que Bzioui [3] s'est aussi intéressé aux liens qui existent entre les méthodes de classification croisée et les modèles probabilistes et propose aussi un modèle probabiliste qui va dans le même sens que celui étudié ici, en adoptant la même démarche probabiliste mais en définissant cette fois-ci un modèle de mélange croisée qui respecte la structure des lignes et des colonnes; en considérant le tableau de données comme un échantillon de taille 1 comme il est fait souvent en traitement statistique d'image. Notre approche diffère de celle de Bzioui [3], dans le sens où comme on va le voir les données du tableau seront considérées comme un échantillon de taille np.

2.1. Notion de mélange croisé

Les données sont toujours fournies sous la forme d'un tableau rectangulaire à n lignes et p colonnes. Rappelons que $\mathbf{I}$ est un ensemble de n individus et $\mathbf{J}$ est un ensemble de p variables.

$\mathbf{I} \times \mathbf{J} = \{(i,j)/i \in \mathbf{I} \text{ et } j \in \mathbf{J}\}$ est un ensemble de np individus-variables; associons à chaque individu-variable la valeur $x_i^j \in \mathbf{R}$ qui correspond à la valeur prise par la variable j pour l'individu i.

On suppose maintenant que l'ensemble $\mathbf{I}$ des individus constitue un échantillon de taille n d'une population Ω; de même on considère que l'ensemble $\mathbf{J}$ des variables constitue un échantillon de taille p d'une population Ω'.

Soit $\mathbf{T} = \mathbf{I} \times \mathbf{J}$ le produit cartésien des deux ensembles $\mathbf{I}$ et $\mathbf{J}$; l'ensemble $\mathbf{T}$ peut être considéré comme un échantillon de taille $n.p$ d'une population $\Omega \times \Omega'$. On peut toujours définir une variable aléatoire $\mathbf{Z}$ qui permet d'associer à chaque couple $(i,j) \in \mathbf{I} \times \mathbf{J}$ la valeur se trouvant à l'intersection de la ligne i avec la colonne j qui est x_i^j.

2.2. Identification d'un mélange «croisé»

Le modèle d'un mélange croisé est le modèle probabiliste qui permet de reconnaître dans une population observée du type $\Omega \times \Omega'$ l'éventuelle présence d'échantillons de lois de probabilité connues. Nous proposons alors le modèle suivant :

Le tableau de données de départ de dimension (n,p) est considéré comme un échantillon $\mathbf{T} = \mathbf{I} \times \mathbf{J}$ de taille $(n.p)$ d'une variable aléatoire à valeur dans $\mathbf{R}$ dont la loi de probabilité admet la fonction de densité :

$$f(x) = \sum_{k=1}^{K} \sum_{m=1}^{M} p_k^m f(x/\lambda_k^m) \tag{2}$$

$$\forall x \in \mathbf{R} \quad \forall k = 1,\ldots,K \quad \text{et } m = 1,\ldots,M \quad 0 \leqslant p_l^m \leqslant 1 \quad \text{et} \sum_{k=1}^{K} \sum_{m=1}^{M} p_k^m = 1.$$

La formule (2) décrit le modèle d'un mélange de type donné $f(.,\lambda_k^m)$ qui est une fonction de densité sur $\mathbf{R}$ appartenant à une famille paramétrée de fonctions de densité

dépendant du paramètre λ_k^m et p_k^m est la probabilité d'apparition de l'observation $f(., \lambda_k^m)$ dans le mélange.

2.3. Problème à résoudre

Problème 1

Le problème consiste à estimer les nombres $\boldsymbol{K}$ *et* $\boldsymbol{M}$ *de composants du mélange et les paramètres inconnus* q_k^m *:*

$(q_k^m = (p_k^m, \lambda_k^m);\ k = 1, \ldots, K$ *et* $m = 1, \ldots, M)$

au vu de l'échantillon $\mathbf{T} = \mathbf{I} \times \mathbf{J}$.

Il s'agit d'un problème d'estimation de paramètres. Nous ne le traiterons pas par l'approche «estimation» du problème mais nous nous concentrerons sur l'approche «classification» pour l'identification d'un mélange «croisé».

2.4. Approche classification

Dans l'approche classification (Scott et Symons [12], Schroeder [11], Bencheikh [2]), on remplace le problème 1 d'estimation par le problème 2 suivant :

Problème 2

Rechercher une partition $P \times Q = \{P_k \times Q^m; k = 1, \ldots, K$ *et* $m = 1, \ldots, M\}$, K *et* M *étant supposés connus, telle que chaque classe* $P_k \times Q^m$ *soit assimilable à un sous-échantillon qui suit une loi* $f(., \lambda_k^m)$.

En suivant l'approche modèle proposé par Schroeder [11] et la représentation de Celeux [4] qui transforme le problème d'optimisation de critère de vraisemblance en un problème d'optimisation de critère de vraisemblance classifiante, on se ramène également, dans le cas de mélange «croisé», à la maximisation de critère de vraisemblance classifiante suivant :

$$\mathbf{VC}(P \times Q, L) = \sum_{k=1}^{K} \sum_{m=1}^{M} \operatorname{Log} R(P_k \times Q^m, \lambda_k^m) \tag{3}$$

où L est le K.M-uple $(\lambda_k^m, k = 1, \ldots, K$ et $m = 1, \ldots, M)$ et $R(P_k \times Q^m, \lambda_k^m)$ est la vraisemblance du sous-échantillon $P_k \times Q^m$ qui suit la loi $f(., \lambda_k^m)$:

$$R(P_k \times Q^m, \lambda_k^m) = \prod_{x \in P_k \times Q^m} f(x/\lambda_k^m).$$

Enfin le critère de vraisemblance classifiante s'écrit :

$$\mathbf{VC}(P \times Q, L) = \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i \in P_k} \sum_{j \in Q^m} \operatorname{Log} f(x_i^j/\lambda_k^m) \tag{4}$$

On peut remarquer que la résolution du problème (4) correspond exactement à la résolution d'un problème de classification croisée (Govaert [7]). Nous proposons d'utiliser le même algorithme que celui défini pour la classification croisée et rappelé dans le paragraphe 1.3. Cet algorithme, que nous allons reprendre pour l'adapter à notre problème, utilise deux algorithmes voisins l'un de l'autre, et tous deux basés sur le principe des Nuées Dynamiques.

2.5. Algorithme

Le principe de cet algorithme est le suivant : en partant de deux nombres K et M et d'une partition initiale $(P \times Q)^{\circ}$ en K.M classes, l'algorithme construit une suite de partitions-noyaux jusqu'à l'obtention d'une partition stable en appliquant successivement les trois fonctions suivantes :

Une fonction de représentation g définie comme suit :

Cette fonction permet de déterminer les K.M noyaux maximisant le critère (4) :

$$\mathbf{VC}(P \times Q, g(P \times Q)) = \underset{L}{\mathrm{Max}}\ \mathbf{VC}(P \times Q, L).$$

On peut facilement voir que ces noyaux sont les estimateurs du maximum de vraisemblance des paramètres associés aux sous-échantillons $\{P_k \times Q^m; k = 1, \ldots, K$ et $m = 1, \ldots, M\}$.

$$g(P \times Q) = g(\{P_k \times Q^m; k = 1, \ldots, K \text{ et } m = 1, ..., M\}) = \{\lambda_k^m, k = 1, \ldots, K \text{ et } m = 1, \ldots, M\} = L.$$

Une fonction d'affectation h définie comme suit :

Cette fonction permet de déterminer, à Q et L fixés, une partition P de l'échantillon $\mathbf{I}$ améliorant le critère $\mathbf{VC}(P \times Q, L)$. Le critère (4) s'écrit alors :

$$\mathbf{VC}(P \times Q, L) = \sum_{k=1}^{K} \sum_{i \in P_k} \mathrm{Log}\, F(x_i/\lambda_k)$$

$$\text{où } F(x_i/\lambda_k) = \prod_{m=1}^{M} \Big(\prod_{j \in Q^m} f(x_i^j/\lambda_k^m) \text{ et } \lambda_k = (\lambda_k^1 \ldots, \lambda_k^M)$$

On retrouve ainsi la forme du critère de vraisemblance classifiante dans le cas de la classification simple. Les éléments de la classe P_k seront définis comme suit :

$$P_k = \{i \in \mathbf{I}/F(x_i/\lambda_k) \geqslant F(x_i/\lambda_{k'}) \text{ avec } k < k' \text{ en cas d'égalité}\}.$$

Une fonction d'affectation h définie comme suit :

Cette fonction permet de déterminer, à P et L fixés, une partition Q de l'échantillon **J** améliorant le critère :

$$\mathbf{VC}(P \times Q, L) = \sum_{m=1}^{M} \sum_{j \in Q^m} \operatorname{Log} F(x^j/\lambda^m)$$

$$\text{où } F(x^j/\lambda^m) = \prod_{k=1}^{K} \Big(\prod_{i \in P_k} f(x_i^j/\lambda_k^m) \text{ et } \lambda^m = (\lambda_1^m, \ldots, \lambda_K^m)$$

Les éléments de la partition Q de l'échantillon **J** seront déterminés comme suit :

$$Q^m = \{j \in \mathbf{J}/F(x^j/\lambda^m) \geqslant F(x^j/\lambda^{m'}) \text{ avec } m < m' \text{ en cas d'égalité}\}.$$

On peut montrer que sous certaines hypothèses (Govaert [7]), cet algorithme est convergent. On obtient à la convergence une partition $P \times Q$ et une estimation des paramètres λ_k^m. Les proportions p_k^m du mélange sont fournies par les fréquences des classes $P_k \times Q^m$.

3. Modèle associé aux données binaires

3.1. La formule générale

On suppose dans ce modèle que les données initiales forment un échantillon de taille $(n.p)$ d'une variable aléatoire **Z** à valeurs dans $\{0, 1\}$ dont la distribution de probabilité est toujours définie par (2); mais ici $f(., \lambda_k^m) = p(., \lambda_k^m)$ est une distribution de probabilité sur $\{0, 1\}$ appartenant à une famille paramétrée de distributions de probabilités.

3.2. Choix de la famille de distribution

On suppose que la variable aléatoire Z définie précédemment suit une des deux lois de Bernoulli suivantes :

1 avec la probabilité $1 - \varepsilon$ et 0 avec la probabilité ε.

1 avec la probabilité ε et 0 avec la probabilité $1 - \varepsilon$

où $\varepsilon \in]0, \frac{1}{2}[$, c'est-à-dire la loi de Bernoulli de paramètre $(1-\varepsilon)$ et la loi de Bernoulli de paramètre ε. On peut alors écrire :

$$p(x/\lambda_k^m) = \varepsilon^{|x-\lambda_k^m|}.(1-\varepsilon)^{1-|x-\lambda_k^m|}$$

où λ_k^m indique quelle est la distribution retenue :

$\lambda_k^m = 1$ pour la première distribution

$\lambda_k^m = 0$ pour la deuxième distribution.

Les paramètres à estimer sont donc les λ_k^m et la valeur ε.

L'expression du critère devient alors :

$$\mathbf{VC}(P \times Q, L, \varepsilon) = (\mathrm{Log}\frac{\varepsilon}{1-\varepsilon}) \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i \in P_k} \sum_{j \in Q^m} d(x_i^j, \lambda_k^m) + n.p\,\mathrm{Log}(1-\varepsilon) \qquad (5)$$

où $d(x_i^j, \lambda_k^m) = |x_i^j - \lambda_k^m|$.

Donc pour $\varepsilon \in]0, \frac{1}{2}[$, $\mathrm{Log}\,\frac{\varepsilon}{1-\varepsilon}$ est négatif. La maximisation du critère $\mathbf{VC}(P \times Q, L, \varepsilon)$ revient donc à la minimisation du critère $\boldsymbol{W}(P \times Q, L)$ présenté dans le paragraphe 1, ce qui montre l'équivalence des deux approches. Il est facile de voir que la valeur ε maximisant le critère $\mathbf{VC}(P \times Q, L, \varepsilon)$ est $\frac{e}{np}$ où e est la valeur du critère obtenu à la convergence : $e = \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i \in P_k} \sum_{j \in Q^m} |x_i^j - \lambda_k^m|$

On obtient donc une interprétation en termes probabiliste du critère de classification que Govaert [7] avait proposé pour les données binaires.

Ce premier mélange noté M_1 de lois de Bernoulli dépend du paramètre ε qui mesure l'écart d'une classe à son centre, paramètre qui ne dépend ni de la partition en lignes ni de la partition en colonnes ni de la partition en lignes et en colonnes, ce qui dans certaines situations, peut s'avérer irréaliste. (La valeur ε est en quelque sorte l'analogue de la variance commune des classes construites par l'algorithme des centres mobiles).

Pour cela nous proposons d'autres critères où le paramètre peut dépendre de la partition en lignes, de la partition en colonnes et de la partition en lignes et en colonnes. On considère alors trois autres mélanges de Bernoulli (notés respectivement M_2, M_3, M_4) que l'on se propose d'étudier dans le paragraphe suivant :

4. Extension du modèle binaire

4.1. Etude du mélange M_2

On remplace ici la valeur ε par les valeurs ε_k (avec $\varepsilon_k \in]0, \frac{1}{2}[$) qui dépendent de chaque classe en lignes mais qui sont toujours les même pour les classes en colonnes. Le critère de vraisemblance classifiante va alors s'écrire :

$$\mathbf{VC}_2(P \times Q, L, \varepsilon) = \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i \in P_k} \sum_{j \in Q^m} \left\{ \left(\mathrm{Log}\frac{\varepsilon_k}{1-\varepsilon_k} \right) |x_i^j - \lambda_k^m| + \mathrm{Log}(1-\varepsilon_k) \right\} \qquad (6)$$

Fonction de représentation (recherche des λ_k^m et des ε_k)

Les λ_k^m sont les valeurs majoritaires de chaque classe $P_k \times Q^m$. Les valeurs des ε_k seront définis comme suit :

$$\mathbf{VC}_2(P \times Q, L, \varepsilon)$$
$$= \sum_{k=1}^{K} \left\{ \left(\mathrm{Log} \frac{\varepsilon_k}{1-\varepsilon_k} \right) \left(\sum_{m=1}^{M} \sum_{i \in P_k} \sum_{j \in Q^m} |x_i^j - \lambda_k^m| \right)) + p.n_k \,\mathrm{Log}(1-\varepsilon_k) \right\}$$

où n_k est le cardinal de la classe P_k. Posons $e_k = \sum_{m=1}^{M} \sum_{i \in P_k} \sum_{j \in Q^m} |x_i^j - \lambda_k^m|$.

e_k est le nombre de fois où les valeurs majoritaires n'ont pas été prises pour la classe P_k.

$$\mathbf{VC}_2(P \times Q, L, \varepsilon) = \sum_{k=1}^{K} \{(p.n_k - e_k)\mathrm{Log}(1-\varepsilon_k) + e_k.\mathrm{Log}\,\varepsilon_k\} = \sum_{k=1}^{K} \psi(\varepsilon_k).$$

Il faut donc maximiser la fonction ψ.

$\psi'(\varepsilon_k) = \frac{-(p.n_k - e_k)}{1-\varepsilon_k} + \frac{e_k}{\varepsilon_k} = 0$. Le maximum est atteint pour $\varepsilon_k = \frac{e_k}{n_k.p}$ qui appartient bien à l'intervalle $]0, \frac{1}{2}[$, sauf dans le cas très particulier où $e_k = 0$.

Fonction d'affectation (Recherche des classes)

On fixe la partition Q.

Posons $x_i^m = \sum_{j \in Q^m} x_i^j$ et $\mu_k^m = q_m.\lambda_k^m$ (q_m est le cardinal de la classe Q^m).

$$\mathbf{VC}_2(P \times Q, L, \varepsilon)$$
$$= \sum_{k=1}^{K} \sum_{i \in P_k} \left\{ \left(\mathrm{Log} \frac{\varepsilon_k}{1-\varepsilon_k} \right) \left(\sum_{m=1}^{M} |x_i^m - \mu_k^m| \right) + p\,\mathrm{Log}(1-\varepsilon_k) \right\}$$
$$= -\sum_{k=1}^{K} \sum_{i \in P_k} \left\{ \left(\mathrm{Log} \frac{1-\varepsilon_k}{\varepsilon_k} \right) .d(x_i, \mu_k) - A_k \right\}.$$

d est une distance de type L_1 et A_k est la quantité $p\,\mathrm{Log}(1-\varepsilon_k)$ qui dépend de k. On affectera donc l'élément x_i à la classe P_k qui minimise la quantité : $\left(\mathrm{Log} \frac{1-\varepsilon_k}{\varepsilon_k}\right) .d(x_i, \mu_k) - p\,\mathrm{Log}(1-\varepsilon_k)$.

On fixe la partition P :

$$\begin{aligned}
&\mathbf{VC}_2(P \times Q, L, \varepsilon) \\
&\quad = \sum_{m=1}^{M} \sum_{j \in Q^m} \left\{ -\sum_{k=1}^{K} \left(\mathrm{Log} \frac{1-\varepsilon_k}{\varepsilon_k} \right) |x_k^j - \gamma_k^m| + \sum_{k=1}^{K} n_k . \mathrm{Log}(1-\varepsilon_k) \right\} \\
&\quad = -\sum_{m=1}^{M} \sum_{j \in Q^m} d_\varepsilon(x^j, \gamma^m) + A'
\end{aligned}$$

où $x_k^j = \sum_{i \in P_k} x_i^j$ et $\gamma_k^m = n_k . \lambda_k^m$.

d_ε est une distance de type L_1 pondérée par les quantités $\mathrm{Log}\, \frac{1-\varepsilon_k}{\varepsilon_k}$ qui dépendent du vecteur ε et A' est la quantité $p \sum_{k=1}^{K} n_k . \mathrm{Log}(1-\varepsilon_k)$ qui ne dépend pas du point x^j. L'élément x^j sera donc affecté à la classe Q^m qui minimise $d_\varepsilon(x^j, \gamma^m)$.

4.2. Etude du mélange M_3

Dans ce mélange le paramètre ε varie suivant la partition en colonnes. On se place exactement dans les mêmes conditions que dans le mélange M_2. On obtient des résultats symétriques.

4.3. Etude du mélange M_4

Le paramètre ε varie cette fois-ci suivant la partition en lignes et en colonnes, c'est le cas le plus général.

Le critère de vraisemblance classifiante s'écrit :

$$\begin{aligned}
&\mathbf{VC}_4(P \times Q, L, \varepsilon) = \sum_{k=1}^{K} \sum_{m=1}^{M} \\
&\quad \left\{ \left(\mathrm{Log} \frac{\varepsilon_k^m}{1-\varepsilon_k^m} \right) \left(\sum_{i \in P_k} \sum_{j \in Q^m} |x_i^j - \lambda_k^m| \right) + n_k . q_m \, \mathrm{Log}(1-\varepsilon_k^m) \right\} \quad (7)
\end{aligned}$$

Fonction de représentation (recherche des λ_k^m et des ε_k^m)

Les λ_k^m sont toujours les valeurs majoritaires pour les classe $P_k \times Q^m$. Il reste donc à maximiser le critère $\mathbf{VC}_4(P \times Q, L, \varepsilon)$ pour trouver les valeurs des ε_k^m.

Posons : $e_k^m = \sum_{i \in P_k} \sum_{j \in Q^m} |x_i^j - \lambda_k^m|$ où e_k^m est le nombre de fois où la valeur majoritaire n'a pas été prise dans la classe $P_k \times Q^m$.

$$\mathbf{VC}_4(P \times Q, L, \varepsilon)$$
$$= \sum_{k=1}^{K} \sum_{m=1}^{M} \{(n_k.q_m - e_k^m)\,\mathrm{Log}(1-\varepsilon_k^m) + e_k^m.\mathrm{Log}\,\varepsilon_k^m\} = \sum_{k=1}^{K} \sum_{m=1}^{M} \varphi(\varepsilon_k^m)$$

Il faut donc maximiser la fonction $\varphi(\varepsilon_k^m) = (n_k.q_m - e_k^m)\mathrm{Log}(1-\varepsilon_k^m) + e_k^m.\mathrm{Log}\,\varepsilon_k^m$.

Le maximum est atteint pour $\varepsilon_k^m = \frac{e_k^m}{n_k.q_m}$ qui appartient bien à l'intervalle $]0, \frac{1}{2}[$, sauf dans le cas très particulier où $e_k^m = 0$.

Fonction d'affectation (recherche des classes)

On fixe la partition Q :

$$\mathbf{VC}_4(P \times Q, L, \varepsilon)$$
$$= \sum_{k=1}^{K} \sum_{i \in P_k} \left\{ \left(\sum_{m=1}^{M} \left(\mathrm{Log} \frac{\varepsilon_k^m}{1-\varepsilon_k^m} \right) |x_i^m - \mu_k^m| \right) + \sum_{m=1}^{M} q_m \mathrm{Log}(1-\varepsilon_k^m) \right\}$$
$$= -\sum_{k=1}^{K} \sum_{i \in P_k} \left\{ d_{\varepsilon_k}(x_i, \mu_k) - \sum_{m=1}^{M} q_m \mathrm{Log}(1-\varepsilon_k^m) \right\}.$$

Cette fois-ci le second terme dépend de k et aura donc une influence ; on affectera x_i à la classe P_k qui minimise la quantité : $d_{\varepsilon_k}(x_i, \mu_k) - \sum_{m=1}^{M} q_m \mathrm{Log}(1-\varepsilon_k^m)$.

On fixe maintenant la partition P :

$$\mathbf{VC}_4(P \times Q, L, \varepsilon)$$
$$= \sum_{m=1}^{M} \sum_{j \in Q^m} \left\{ \sum_{k=1}^{K} \left(\mathrm{Log} \frac{\varepsilon_k^m}{1-\varepsilon_k^m} \right) (|x_k^j - \gamma_k^m|) + \sum_{k=1}^{K} n_k.\mathrm{Log}(1-\varepsilon_k^m) \right\}$$
$$= -\sum_{m=1}^{M} \sum_{j \in Q^m} \left\{ d_{\varepsilon^m}(x^j, \gamma^m) - \sum_{k=1}^{K} n_k.\mathrm{Log}(1-\varepsilon_k^m) \right\}$$

On se retrouve là aussi dans la même situation que dans le cas précédent. Le second terme dépend de m et aura donc une influence; on affectera x^j à la classe Q^m qui minimise la quantité :

$$d_{\varepsilon^m}(x^j, \gamma^m) - \sum_{k=1}^{K} n_k.\mathrm{Log}(1-\varepsilon_k^m).$$

Nous venons de voir que l'identification d'un mélange de Bernouilli avec le même paramètre pour toutes les classes correspond au critère de classification

optimisé par la méthode CROBIN. La généralisation de ce modèle en considérant différents paramètres permet de proposer un nouvel algorithme utilisant des distances adaptatives que nous appellerons algorithme CROBIN adaptatif qui est composé de quatre variantes : La variante 1 associée au mélange M_1 (CROBIN), la variante 2 correspondant au mélange M_2, la variante 3 au mélange M_3 et la variante 4 correspondant au mélange M_4.

On propose maintenant de comparer les résultats obtenus en appliquant successivement les quatre variantes (V_1, V_2, V_3, V_4) de l'algorithme CROBIN adaptatif sur des données réelles et des données simulées.

5. Application et comparaison des méthodes

Pour réduire l'influence de la partition d'initialisation des algorithmes sur les résultats, nous avons opté pour la démarche suivante :

Première stratégie : On applique la variante 1 de l'algorithme CROBIN adaptatif en demandant 20 tirages, puis on applique successivement les variantes 2, 3 et 4 en les initialisant par le meilleur couple de partitions obtenue par la variante 1. On compare les résultats obtenus par rapport au couple de partitions d'initialisation dans le cas de données réelles et au couple de partitions ayant servi à la simulation dans le cas de données simulées.

Deuxième stratégie : On applique la variante 2 en demandant 20 tirages. Les variantes 3 et 4 sont initialisées par le meilleur couple de partitions obtenu par la variante 2.

Dans le cas de données réelles, on compare là aussi le couple de partitions des variantes 3 et 4 au meilleur obtenu dans la variante 2 (i.e au couple de partitions initiales adopté pour les variantes 3 et 4). Dans le cas des données simulées, on compare le couple de partitions obtenu par les variantes 3 et 4 à celui ayant servi à la simulation.

Troisième stratégie : On applique la variante 3 en demandant 20 tirages. La variante 4 est initialisée par le meilleur couple de partitions obtenue par la variante 3. On compare le couple de partitions obtenu par la variante 4 au meilleur couple de la variante 3 pour les données réelles et à celui ayant servi à la simulation pour les données simulées.

Quatrième stratégie : On applique la variante 4 de l'algorithme en demandant 20 tirages. On compare ce couple de partitions à celui ayant servi à la simulation (pour les données simulées).

5.1. Données réelles

On a appliqué l'algorithme CROBIN adaptatif sur les données nommées : « Information and store choice » traitées par Goldstein et Dillon [6] en demandant trois classes en lignes et deux classes en colonnes. Il s'agit de 412 individus caractérisés

par 4 variables binaires. Le tableau des données étudié est donc un tableau binaire à 412 lignes et 4 colonnes.

Les valeurs du tableau ci dessous indiquent pour chaque cas le nombre d'éléments en lignes et le nombre d'éléments en colonnes qui ont été classés différemment. Par exemple la valeur (19,1) obtenue par la variante 4 indique qu'il y a 19 éléments en ligne et 1 élément en colonne qui ont été classés différemment par rapport au couple de partitions d'initialisation.

(a) Résultats obtenus par la première stratégie			
	V_2	V_3	V_4
Nombre d'éléments classés différemment :	(34,1)	(34,1)	(19,1)
Pourcentage d'éléments mal classés :	(8.25 %, 25 %)	(8.25 %, 25 %)	(4.61 %, 25 %)
(b) Résultats obtenus par la deuxième stratégie			
		V_3	V_4
Nombre d'éléments classés différemment :		(6,0)	(4,0)
Pourcentage d'éléments mal classés :		(1.45 %, 0 %)	(0.97 %, 0 %)
(c) Résultats obtenus par la troisième stratégie			
			V_4
Nombre d'éléments classés différemment :			(15,2)
Pourcentage d'éléments mal classés :			(3.64 %, 50 %)

Remarque : Les variantes 2, 3 et 4 arrivent en général à modifier le couple de partitions obtenue par la variante 1 (CROBIN) , la comparaison des partitions obtenues dans les trois approches est délicate; il est donc probable que l'algorithme CROBIN adaptatif ait ainsi mis en évidence une structure difficile à découvrir. Pour cette raison nous proposons de nous appuyer sur des modèles probabilistes pour simuler des données sur lesquelles on a pu approfondir notre comparaison.

5.2. Données simulées

Pour ces données on a réalisé un programme de simulation de données binaires, ce dernier construit un tableau de données plus ou moins proche d'un tableau idéal structuré en blocs de 1 et 0. Pour cela il faut fournir les nombres de classes des partitions en lignes et en colonnes, le nombre d'éléments de chacune des classes des deux partitions, le tableau des valeurs idéales et enfin la probabilité avec laquelle le tableau ainsi défini va s'approcher du tableau idéal. Nous proposons aussi quatre variantes pour cet algorithme : la même probabilité de tirage ε, probabilité différente pour chaque classe en ligne ε_k, probabilité différente pour chaque classe en colonne ε^m et enfin probabilité différente pour chaque couple de classe ε_k^m.

Chaque valeur du tableau est obtenue en faisant un tirage de Bernoulli des deux nombres 0 et 1 avec les probabilités ε et $(1-\varepsilon)$, si la valeur idéale est 1; $(1-\varepsilon)$ et ε si la valeur idéale est 0 ($\varepsilon \in]0, \frac{1}{2}[$); à la fin du programme, le tableau est permuté de manière aléatoire en lignes et en colonnes.

A l'aide du programme précédent, on a simulé quatre tableaux binaires de dimensions 100×30 que l'on nomme par simul1, simul2, simul3 et simul4. Les paramètres ayant servi à la simulation de ces tableaux sont :

$$\begin{array}{l} K = 3, M = 3 \\ n_1 = 33, n_2 = 33, n_3 = 34 \\ q_1 = 10, q_2 = 10, q_3 = 10 \end{array}$$

Les tableaux des valeurs idéales $\{\lambda_k^m, k = 1, \ldots, K$ et $m = 1, \ldots, M\}$ associés à simul1, simul2, simul3 et simul4 sont respectivement :

$$\begin{array}{cccc} 101 & 110 & 100 & 001 \\ 010 & 001 & 111 & 100 \\ 000 & 000 & 100 & 111 \end{array}$$

Simul1 (même probabilité de tirage pour tout les éléments : $\varepsilon = 0,10$)

Simul2 (probabilité de tirage différente pour chaque classe en ligne : $\varepsilon_1 = 0,10$; $\varepsilon_2 = 0,20$; $\varepsilon_3 = 0,30$)

Simul3 (probabilité de tirage différente pour chaque classe en colonne : $\varepsilon^1 = 0,20$; $\varepsilon^2 = 0,10$; $\varepsilon^3 = 0,40$)

Simul4 (probabilité de tirage différente pour chaque classe en ligne et en colonne : $\varepsilon_1^1 = 0,15$; $\varepsilon_1^2 = 0,25$; $\varepsilon_1^3 = 0,05$; $\varepsilon_2^1 = 0,10$; $\varepsilon_2^2 = 0,40$; $\varepsilon_2^3 = 0,20$; $\varepsilon_3^1 = 0,45$; $\varepsilon_3^2 = 0,30$; $\varepsilon_3^3 = 0,35$)

Le tableau ci-dessous résume l'ensemble des résultats obtenus en appliquant l'algorithme CROBIN adaptatif aux quatre fichiers simulés et en demandant 3 classes en lignes et 3 classes en colonnes.

Les valeurs du tableau indiquent pour chaque cas le nombre d'éléments en lignes et le nombre d'éléments en colonnes qui ont été classés différemment. Par exemple la valeur (8,4) obtenue pour Simul4 dans la première stratégie indique qu'il

(a) Résultats obtenus par la première stratégie					
		V_1	V_2	V_3	V_4
Nombre d'éléments classés différemment :	Simul1	(2,2)	(2,2)	(2,2)	(2,2)
	Simul2	(10,3)	(3,2)	(3,2)	(4,3)
	Simul3	(10,3)	(3,2)	(3,2)	(4,3)
	Simul4	(8,4)	(8,3)	(8,3)	(4,2)
Pourcentage d'éléments mal classés :	Simul1	(2%,6.6%)	(2%,6.6%)	(2%,6.6%)	(2%,6.6%)
	Simul2	(10%,10%)	(3%,6.6%)	(3%,6.6%)	(4%,10%)
	Simul3	(10%,10%)	(3%,6.6%)	(3%,6.6%)	(4%,10%)
	Simul4	(8%,13.3%)	(8%,10%)	(8%,10%)	(4%,6.6%)
(b) Résultats obtenus par la deuxième stratégie					
			V_2	V_3	V_4
Nombre d'éléments classés différemment :	Simul1		(2,2)	(2,2)	(2,2)
	Simul2		(1,1)	(1,1)	(1,1)
	Simul3		(2,3)	(2,1)	(2,2)
	Simul4		(2,1)	(2,1)	(2,2)
Pourcentage d'éléments mal classés :	Simul1		(2%,6.6%)	(2%,6.6%)	(2%,6.6%)
	Simul2		(1%,3.3%)	(1%,3.3%)	(1%,3.3%)
	Simul3		(2%,10%)	(2%,3.3%)	(2%,6.6%)
	Simul4		(2%,3.3%)	(2%,3.3%)	(2%,6.6%)
(c) Résultats obtenus par la troisième stratégie					
				V_3	V_4
Nombre d'éléments classés différemment :	Simul1			(3,4)	(7,2)
	Simul2			(2,2)	(5,2)
	Simul3			(1,0)	(1,0)
	Simul4			(2,1)	(3,0)
Pourcentage d'éléments mal classés :	Simul1			(3%,13.3%)	(7%,6.6%)
	Simul2			(2%,6.6%)	(5%,6.6%)
	Simul3			(1%,0%)	(1%,0%)
	Simul4			(2%,3.3%)	(3%,0%)
(d) Résultats obtenus par la quatrième stratégie					
					V_4
Nombre d'éléments classés différemment :	Simul1				(6,3)
	Simul2				(7,4)
	Simul3				(7,4)
	Simul4				(0,0)
Pourcentage d'éléments mal classés :	Simul1				(6%,10%)
	Simul2				(7%,13.3%)
	Simul3				(7%,13.3%)
	Simul4				(0%,0%)

y a 8 éléments en ligne et 4 éléments en colonne qui ont été classés différemment par rapport au couple de partitions ayant servi à la simulation du fichier Simul4.

On remarque que sur les deux types de données, l'algorithme CROBIN adaptatif modifie plus ou moins le couple de partitions obtenue par la variante 1, les valeurs des différents coefficients de pondérations sont raisonnables (pas d'écart exagéré entre les différentes valeurs) pour les quatre cas de pondérations; en revanche l'algorithme CROBIN adaptatif présente un grand avantage pour les données simulées, car chaque variante de l'algorithme permet de donner le couple de partitions le plus proche de celui qui a servi à la simulation du tableau de données suivant le modèle de la variante en question (voir tableau ci-dessus).

L'algorithme CROBIN adaptatif permet d'améliorer les résultats obtenus par l'algorithme CROBIN dans lequel, on ne tient pas compte des pondérations; c'est-à-dire qu'on donne à toutes les variables et les variables-classes le même poids, ce qui statistiquement peut être vu comme un défaut.

6. Conclusion

Dans ce travail, nous avons établi comment la méthode CROBIN (classification croisée sur données binaires) peut s'interpréter comme l'approche classification associée à un mélange de distributions de Bernoulli avec le même paramètre pour toutes les classes. La généralisation de ce modèle en considérant différents paramètres permet de proposer un nouvel algorithme de classification croisée binaire utilisant des distances adaptatives. Ces dernières correspondent respectivement aux mélanges étudiés précédemment.

Le lien qui existe entre la méthode de classification croisée et le modèle probabiliste proposé dans ce papier permet aussi d'expliquer les bons résultats que Govaert [7] avait obtenus à l'aide de cet algorithme sur des données simulées qui justement suivaient ce modèle.

Bibliographie

[1] Y. BENCHEIKH, Classification automatique et modèles, Thèse, Université de Metz, 1992.

[2] Y. BENCHEIKH, Classification croisée et modèles, Rairo operations research, vol. 33.4, p. 525-541, 1999.

[3] M. BZIOUI, Classification croisée et modèles, Thèse, Université de Metz, 1999.

[4] G. CELEUX, Classification et modèles, Rev. statist. Appl., n° 4, p. 43-58, 1988.

[5] E. DIDAY, Nouvelles méthodes et nouveaux concepts en classification automatique et reconnaissance des formes, Thèse d'Etat Université Paris 6, 1972.

[6] M. GOLDSTEIN et W.R DILLON, Discrete discriminant analysis, John Wiley & Sons, New-York (1978).

[7] G. GOVAERT, Classification croisée, Thèse de Doctorat d'Etat, Université Pierre et Marie Curie, Paris 6, 1983.

[8] G. GOVAERT, Classification binaire et modèles, Rev. Statistique Appliquée, 1990 XXXVIII (1), 67-81.

[9] G. GOVAERT, Simultaneous clustering of rows and columns, Control and cybernetics vol. 24 (1995), N° 4.

[10] M. NADIF et F MARCHETTI, Classification de données qualitatives et modèles, Rev. statistique Appliquée, 1993, XLI (1), 55-69.

[11] A. SCHROEDER, Reconnaissance des composants d'un mélange, Thèse de Doctorat de 3ème cycle, Université de Paris 6, 1974.

[12] A. SCOTT et SYMONS, Clustering methods based on likelihood ratio criteria, Biometrics 27, 387-397, 1971.