Classification non Supervisée de Données Multidimensionnelles par les Processus Ponctuels Marqués

HENNI Khadidja* — ALATA Olivier *** — ZAOUI Lynda* — ELIDRISSI Abdellatif ** —MOUSSA Ahmed **

* Laboratoire LSSD, Département d'informatique, Université des Sciences et Technologie 'Mohamed Boudiaf 'USTO-MB, Oran-Algérie
Email :{ khadidja.henni, lynda.zaoui}@univ-usto.dz
** Laboratoire LabTIC, ENSAT, Université Abdelmalek Essaadi, BP 1818, Tanger-Maroc
Email : amoussa@uae.ac.ma
*** Laboratoire Hibert Curien, UMR 5516, Université Jean Monnet, Saint Etienne-France
Email : olivier.alata@univ-stetienne.fr

RÉSUMÉ. Cet article décrit un nouvel algorithme non supervisé de classification des données multidimensionnelles. Il consiste à détecter les prototypes des classes présentes dans un échantillon et à appliquer l'algorithme KNN pour la classification de toutes les observations. La détection des prototypes des classes est basée sur les processus ponctuels marqués, c'est d'une part une adaptation de la méthode de Métropolis-Hasting-Green qui génère des mouvements manipulant les objets du processus (naissance, mort…) et d'autre part une modélisation de Gibbs qui introduit la fonction de potentiel matérialisant les interactions du processus en termes d'énergie. Plusieurs expérimentations ont été réalisées sur des données ponctuelles multidimensionnelles où les classes sont non linéairement séparables et des données réelles issues des puces à ADN. Une comparaison avec des méthodes de classification existantes a permis de montrer l'efficacité de ce nouvel algorithme.

ABSTRACT. We present in this work a new unsupervised clustering algorithm, it detects the prototypes of the classes and uses KNN algorithm to classify all available observations. The prototype class detection is based on marked point processes. It is an adaptation of : the Metropolis-Hasting-Green method that generate several movements to manipulate the objects processes (birth, death ..); the Gibbs theory that models the potential function of the process.
Several experiments were performed on multidimensional data with non linearly separable classes and on real microarray data. A comparison with existing classification methods has shown the effectiveness of this new algorithm.

MOTS-CLÉS : Classification automatique, Processus ponctuels marqué, Détection des modes, Metropolis Hasting-Green.

KEYWORDS: Automatic classification, Marked point process, Mode detection, Metropolis Hasting-Green.

1. Introduction

La classification automatique non supervisée de données multidimensionnelles consiste à attribuer une classe à chaque objet sans aucune information *a priori*. Il existe une très large famille de méthodes dédiées à la classification automatique telles que les méthodes de partitionnement (k-means) [1], où le regroupement des observations en classes se fait sur des considérations géométriques nécessitant la définition d'une mesure de proximité entre observations et un choix judicieux de variables de calcul de distance [2], et les méthodes probabilistes qui s'appuient sur l'analyse de la densité de probabilité de la population [2]. Parmi les méthodes probabilistes, nous pouvons citer :

- l'algorithme de mélanges de distributions qui traduit l'homogénéité par le fait que les observations doivent être issues d'une même distribution. Bien qu'il soit performant, sa principale limite est sa forte dépendance aux valeurs initiales [2], ce qui le rend instable.

- l'algorithme DBSCAN [3] qui est parmi les premiers à employer la notion de densité dans la classification automatique. Ici, l'homogénéité est définie par le fait que les observations sont localisées sur une même région de forte densité. Cet algorithme dépend essentiellement des paramètres mesurant la distance de voisinage et la densité minimale. Plusieurs variantes de cet algorithme ont été proposées [4], mais elles partagent toutes la même faiblesse face au problème des classes déséquilibrées et la dépendance aux paramètres initiaux.

-Les champs de Markov, employés dans la détection des modes de classes à partir des données multidimensionnelles [5]. Leurs principales limites sont d'une part le nombre de variables aléatoires qui doit être fixé et qui est une connaissance *a priori* forte que nous n'avons pas dans la plupart des cas et d'autre part, les contraintes qui matérialisent les interactions locales sont difficiles à prendre en compte.

Pour résoudre ce type de problèmes notamment dans un contexte probabiliste, nous proposons dans cet article l'adaptation des processus ponctuels marqués (PPM) pour la classification automatique non supervisée des données multidimensionnelles. L'idée clé est de considérer les distributions de données comme une réalisation d'un PPM où les marques sont des objets géométriques (hypersphères) : les objets (point + marque) doivent se localiser dans les régions à forte concentration de données et l'interaction de ces derniers produit des objets complexes qui sont les modes de classes que nous recherchons. Ensuite, les observations non-prototypes sont affectées aux classes correspondantes par l'algorithme KNN [6].

L'article est organisé comme suit. Dans la première section, nous présentons les PPM, leur impact sur notre problématique et le modèle que nous proposons pour ces processus. Dans la deuxième section, nous décrivons l'algorithme de simulation des PPM proposé et nous procédons à l'ajustement des paramètres du modèle. Dans la troisième section, nous présentons les expérimentations réalisées respectivement sur des données simulées et des données réelles. Les résultats de ces expérimentations sont

comparés à ceux obtenus par d'autres algorithmes probabilistes de classification non supervisée. Nous concluons cet article en proposant des perspectives à ce travail.

2. Modèle pour la détection des modes de classes

2.1. Les processus ponctuels marqués :

Les processus ponctuels décrivent des distributions de points dans l'espace, ils sont construits sur des contraintes définies préalablement, le plus simple étant le processus ponctuel de Poisson [7]. Si le processus impose des contraintes d'interactions entre des points de voisinage, il est dit Markovien [7]. Les processus ponctuels marqués sont des processus dont les configurations sont des ensembles d'objets ayant une position et une marque géométrique [8][9]. Ils sont définis par leur densité par rapport à une mesure de Poisson de référence.

La définition d'un processus ponctuel marqué nécessite la définition d'un processus ponctuel simple, fini et composé d'un ensemble de points considérés comme les positions des marques. Ces points sont choisis aléatoirement dans un sous-espace complet $\chi \subseteq \mathbb{R}^M$ (χ est un sous-espace fini : $\mu(\chi) < \infty$ où μ est la mesure de Lebesgue).

La fonction de densité de probabilité du processus peut s'exprimer sous forme d'une distribution de Gibbs :

$$f(x) \propto e^{U(x)} \quad (1)$$

où $U(x)$ est l'énergie du processus [10].

2.2. Application des PPM pour la détection des modes de classes :

L'objectif de cet article est de proposer un algorithme de classification automatique de données multidimensionnelles. Cet algorithme est composé de deux étapes, la détection des modes de classes en se basant sur des considérations statistiques et la classification de toutes les observations par le KNN [6].

La répartition aléatoire des observations dans l'espace engendre la constitution des régions à concentration élevée d'observations; ces régions forment les modes de classes recherchées. Ces modes sont des nuages de points de différentes formes, caractérisés par une densité élevée (Figure 1) et composés d'observations prototypes.

La détection des modes de classes peut être vue comme un marquage de régions à forte concentration dans l'espace, ce qui nous a amené à utiliser les processus ponctuels marqués qui sont généralement utilisés pour marquer et extraire des objets à partir des images [8] [9] [11]. L'hypothèse clé consiste à considérer les observations prototypes comme des réalisations d'un PPM où le processus est un ensemble de configurations aléatoires d'objets, ces objets se placent sur les régions à forte concentration de données

et leurs unions forment des objets complexes qui épousent les formes des modes de classes recherchées (Figure 2).

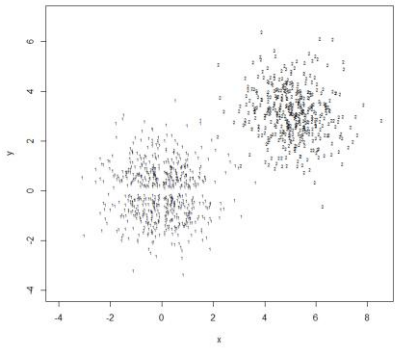

Figure 1. ***Ensemble d'observations.***

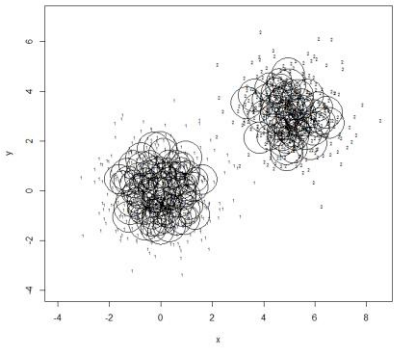

Figure 2. ***Modes des classes détectés.***

Afin d'adapter les PPM au problème d'analyse de données que nous traitons, nous définissons:

- Un champ d'observations Y: il correspond à l'ensemble des données multidimensionnelles observées, $y = \{y_q\}_{q=1,\dots,Q}$ où Q est le nombre d'observations et $y_q = [y_{q,1}, \dots, y_{q,M}]^T \in \chi \subseteq \mathbb{R}^M$où M est la dimension de l'espace.
- Un PPM X: une réalisation du processus est définie par un ensemble d'hypersphères $x = \{x_l\}_{l=1,\dots,n(x)}$. Chaque hypersphère x_l est définie par un centre $c_l \in y$ (observation) et un rayon $r_l \in [r_{min}, r_{max}]$ (il définit les objets de voisinage), $n(x)$ est le nombre d'hypersphères dans la configuration x. Une hypersphère (ou objet) est notée $x_l(c_l, r_l)$, la configuration de points dans x est notée $p_x = \{c_l\}_{l=1,\dots,n(x)}$.

Nous supposons qu'un sous ensemble d'observations est une réalisation des points du PPM X que nous avons définie précédemment. Ce sous-ensemble sera l'ensemble des points initiaux des prototypes des classes. Dans l'étape de simulation, nous proposons de déterminer la réalisation du PPM en sélectionnant des observations $c_l \in y, l = 1, \dots, n(\hat{x})$ associées à des marques qui maximisent la fonction de densité de probabilité f du processus X, telle que la configuration optimale d'objets, $\hat{x}$, est :

$$\hat{x} = argmax_{x, p_x \subseteq y} f(x) \quad (2)$$

Contrairement à la définition classique des PPM, l'algorithme proposé réduit l'ensemble des positions (points); il ne tire pas les valeurs contenues dans p_x du sous-espace χ, mais il se limite aux observations dans l'ensemble y. Ceci lui permet de

converger plus rapidement puisqu'il ne traite pas tous les points de l'espace, mais uniquement les points d'intérêt : 'les observations'.

L'algorithme sélectionne un ensemble optimal d'observations; ces points optimaux seront localisés dans les régions à forte densité (les modes de classes). Les marques vont aider à estimer le nombre de modes. L'intersection des hypersphères définit les régions connectées qui épousent les formes des classes dans les distributions d'origines (voir figure 2).

2.3. Modèle proposé :

L'énergie d'un processus ponctuel marqué est la somme de deux termes [10], l'énergie d'attache aux données et l'énergie d'interaction:

$$U(x|\theta) = U_d(x|\theta) + U_i(x|\theta) \quad (3)$$

où θ est le vecteur de paramètres du modèle. L'énergie d'attache aux données $U_d(x|\theta)$ est l'énergie externe qui représente la relation entre les objets et les observations.

L'énergie d'interaction $U_i(x|\theta)$ est une énergie interne qui modélise les interactions entre les objets.

L'énergie d'attache aux données $U_d(x|\theta)$ est définie par le modèle détecteur [8], elle s'exprime par l'équation suivante:

$$U_d(x|\theta) = -\sum_{x_i \in X} v(x_i) \quad (4)$$

où $v(x_i)$ est la fonction de potentiel associée à l'hypersphère $x_i(c_i, r_i)$.

Afin de résoudre notre problème, l'approche proposée consiste à placer les objets dans les régions à forte concentration de données. De ce fait, cette fonction devrait favoriser l'acceptation des objets bien positionnés. Nous considérons l'objet $x_i(c_i, r_i)$ bien positionné s'il couvre au moins n_{min} observations et sa densité $d(x_i)$ est supérieure à d_{min} ($n_{min}, d_{min} \in \theta$ avec $n(x_i) > n_{min}$, $d(x_i) > d_{min}$ et $n(x_i)$ le nombre d'observations couvertes par l'objet). De ce fait, nous proposons:

$$\mathrm{v}(x_i) = \begin{cases} n(x_i) & si\ n(x_i) > n_{min}\ \ et\ \ d(x_i) > d_{min} \\ -n(x_i) - v_{max} & sinon \end{cases} \quad (5)$$

où v_{max} est une grande valeur qui réduit les chances d'acceptation des objets mal positionnés.

L'énergie d'interaction est modélisée par l'équation suivante :

$$U_i(x|\theta) = n(x)\log\beta - |co(x)|\log\gamma + n_v \log\delta \quad (6)$$

où β est le paramètre d'intensité. L'énergie d'interaction est définie par le processus par composantes connexes et le *processus par paire de points.*

Le processus par composantes connexes permet de trouver le nombre optimal des composantes connexes, qui correspond au nombre de régions à forte concentration de

données. A cet effet, notre objectif est de faire un lien entre la création des composantes connexes $co(x)$ dans une configuration x avec la dispersion des observations dans l'espace. $\gamma > 0$ est le paramètre d'interaction. Chaque composante connexe $co_i \in co(x), i = 1, \dots, |co(x)|$ à une contribution de : $n(co_i)log\beta - log\gamma$ où $n(co_i)$ est le nombre d'hypersphères contenues dans co_i(pour plus de détails voir [12][13]).

Le processus par paire de points permet de pénaliser le recouvrement des hypersphères. Il se base sur la définition de voisinage suivante, pour $x_i \in x$ et $x_j \in x, i \neq j$:

$$x_i \sim x_j \; si \; d(c_i, c_j) < \frac{r_i + r_j}{s} \tag{7}$$

La valeur du coefficient de voisinage s a été fixée expérimentalement de façon à obtenir une mesure de voisinage ("hard-core"), qui assure une stabilité rapide du processus, mais qui ne soit pas sévère pour ne pas défavoriser le chevauchement et l'intersection des objets.

Le potentiel d'interaction du second ordre peut s'écrire :

$$\phi(x_i, x_j) = \begin{cases} log\delta & si \; x_i \sim x_j \\ 0 & sinon \end{cases} \tag{8}$$

La contribution de ce terme dans l'énergie d'interaction est $\sum_{(x_i,x_j),i<j} \phi(x_i, x_j) = n_v \log \delta$ avec n_v le nombre des relations de voisinage dans la configuration x et $\delta \in [0,1]$. Nous choisissons δ proche de 0 pour pénaliser fortement les recouvrements des objets.

La définition des énergies d'attache aux données et d'interaction est liée au vecteur de paramètres $\theta = \{n_{min}, d_{min}, \beta, \gamma, \delta, r_{min}, r_{max}\}$, où r_{min}et r_{max} sont respectivement les rayons minimum et maximum des hypersphères.

Nous présentons dans la prochaine section l'algorithme de détection des modes et l'ajustement des paramètres du modèle.

3. Algorithme proposé

L'algorithme de classification automatique proposé dénommé 'MDMPP-KNN' pour (Mode Detection using Marked Point Process-K Nearest Neighbors) est la succession de deux algorithmes. L'algorithme MDMPP qui détecte les modes des classes et qui fait l'objet de ce papier, il partitionne l'ensemble y en observations prototypes assignées à des classes et d'autres non-prototypes sans assignation. Une version améliorée de l'algorithme « KNN » pour la classification des observations non-prototypes, en commençant par affecter les observations les plus proches des prototypes des classes correspondantes [6].

3.1. Algorithme de détection des modes de classes MDMPP :

Cet algorithme est une simulation du processus ponctuel marqué que nous avons modélisé dans la section précédente. Il existe plusieurs algorithmes de simulation des PPM. Nous avons opté pour l'adaptation de la technique RJMCMC [14] qui ne se limite pas à de simples mouvements de naissance et de mort, mais permet d'ajouter d'autres mouvements. C'est un processus de naissance avec rejets (les cas de naissance qui ne remplissent pas les critères initiaux ne sont pas autorisés), ce qui permet de cesser la croissance des objets, favorisant ainsi une stabilisation de l'algorithme.

MDMPP est un algorithme itératif qui génère une configuration d'objets à chaque itération : MDMPP choisit aléatoirement un des mouvements (naissance, mort, déplacement ou changement du rayon); une nouvelle configuration est donc proposée suivant le mouvement choisi. L'algorithme calcule un rapport dit « Green's ratio » GR [14] ; ce rapport permet de calculer la probabilité d'acceptation/refus de la nouvelle configuration. Les GR de chaque mouvement sont donnés par rapport aux probabilités du mouvement choisi ($P_{naissance}, P_{mort}$), la densité de l'ancienne et la nouvelle configuration ($f(x), f(\hat{x})$), le nombre d'objets dans la configuration ($n(x)$) et la mesure de l'espace $\mu(\chi)$.

Notons $x^i, i > 0$, la configuration des objets à la $i^{éme}$ itération de l'algorithme MDMPP. $p_{naissance}$, p_{mort}, $p_{déplacement}$, $p_{changement\ r}$ sont les probabilités de choix des mouvements : naissance, mort, déplacement et changement de rayon. Avant le début de la simulation, MDMPP génère une configuration initiale $x^0 = \{x_l\}_{l=1,\dots,n(x^0)}$ qui est un ensemble d'objets qui couvrent toutes les observations (Figure 3.a) :

Fonction d'initialisation :

MDMPP répète les étapes suivantes jusqu'à ce que toutes les observations soient couvertes :

a. sélectionne une observation non couverte $\mathbf{y_l}$
b. Génère un objet $\mathbf{x_l} \in \mathbf{x^0}$ de centre $\mathbf{c_l} = \mathbf{y_l}$ et de rayon maximal $\mathbf{r_l} = \mathbf{r_{max}}$.
c. Ajoute l'objet $\mathbf{x_l}$ à la configuration $\mathbf{x^0}$, $\mathbf{x^0} = \mathbf{x^0} \cup \{\mathbf{x_l}\}$.

Après l'initialisation, MDMPP estime les paramètres du modèle et passe à la simulation du PPM en se basant sur le modèle présenté dans la section 2.3, la simulation est une adaptation de RJMCMC :

Fonction de simulation :

MDMPP répète les étapes suivantes, jusqu'à la stabilisation de la moyenne du nombre d'objets dans le processus et la moyenne de son énergie :

a- Choix du mouvement : à la $(\mathbf{i+1})^{\mathbf{éme}}$ itération, l'algorithme commence par choisir aléatoirement un des mouvements (naissance, mort,

déplacement et changement de rayon).

b- Simulation du mouvement : en fonction du mouvement choisi, l'algorithme fait les étapes suivantes :

Naissance : ce mouvement se déroule en trois étapes :

1. Création d'un objet $\boldsymbol{\omega(c_\omega, r_\omega)}$ où $\mathbf{c_\omega}$ et $\mathbf{r_\omega}$ sont choisis uniformément dans l'ensemble d'observations $\mathbf{y}$ et dans $[\mathbf{r_{min}}, \mathbf{r_{max}}]$ respectivement ($\mathbf{c_\omega \sim U_y}$ **et** $\mathbf{r_\omega \sim U_{(r_{min};r_{max})}}$).
2. Ajout de $\boldsymbol{\omega}$ dans la configuration $\mathbf{x^i}$, la nouvelle configuration devient $\tilde{\mathbf{x}}_{\mathbf{naissance}} = \mathbf{x^i} \cup \{\boldsymbol{\omega}\}$.
3. Calcul du **GR**correspondant

$$\mathbf{GR_{naissance}} = \frac{\mathbf{p_{mort}}}{\mathbf{p_{naissance}}} \frac{\mathbf{f(\tilde{x}_{naissance})}}{\mathbf{f(x)}} \frac{\boldsymbol{\mu(\chi)}}{\mathbf{n(x)+1}} \quad (9)$$

Mort : si la configuration $\mathbf{x^i}$contient au moins un objet, la simulation de ce mouvement se déroulera en trois étapes :

1. Sélection aléatoire d'un objet $\boldsymbol{\omega}$ dans la configuration $\mathbf{x^i}$ ($\boldsymbol{\omega \sim U_{(x^i)}}$).
2. Suppression de $\boldsymbol{\omega}$ de la configuration $\mathbf{x^i}$, la nouvelle configuration devient $\tilde{\mathbf{x}}_{\mathbf{mort}} = \mathbf{x^i} \backslash \{\boldsymbol{\omega}\}$.
3. Calcul du **GR** correspondant.

$$\mathbf{GR_{mort}} = \frac{\mathbf{p_{naissance}}}{\mathbf{p_{mort}}} \frac{\mathbf{f(\tilde{x}_{mort})}}{\mathbf{f(x)}} \frac{\mathbf{n(x)}}{\boldsymbol{\mu(\chi)}} \quad (10)$$

Déplacement : si la configuration $\mathbf{x^i}$contient au moins un objet, la simulation de ce mouvement se déroulera en trois étapes :

1. Sélection aléatoire d'un objet $\boldsymbol{\omega}$ dans la configuration $\mathbf{x^i}$ et d'un nouveau centre $\mathbf{c_{\tilde{\omega}}}$ de l'ensemble d'observations $\mathbf{y}$ ($\boldsymbol{\omega \sim U_{(x^i)}}$ **et** $\mathbf{c_{\tilde{\omega}} \sim U_{(y)}}$). Le nouvel objet devient $\tilde{\boldsymbol{\omega}}$ ($\mathbf{c_{\tilde{\omega}}}, \mathbf{r_\omega}$).
2. Remplacement de $\boldsymbol{\omega}$ par $\tilde{\boldsymbol{\omega}}$ dans la configuration $\mathbf{x^i}$, la nouvelle configuration devient $\tilde{\mathbf{x}}_{\mathbf{deplacement}} = \{\mathbf{x^i} \backslash \{\boldsymbol{\omega}\}\} \cup \{\tilde{\boldsymbol{\omega}}\}$.
3. Calcul du **GR**correspondant.

$$\mathbf{GR_{déplacement}} = \frac{\mathbf{f(\tilde{x})}}{\mathbf{f(x)}} \quad (11)$$

Changement du rayon : si la configuration $\mathbf{x^i}$ contient au moins un objet, la simulation de ce mouvement se déroulera en trois étapes :

1. Sélection aléatoire d'un objet $\boldsymbol{\omega}$ de la configuration $\mathbf{x^i}$ et d'un nouveau rayon $\mathbf{r_{\tilde{\omega}}}$ de $[\mathbf{r_{min}}, \mathbf{r_{max}}]$ ($\boldsymbol{\omega \sim U_{(x^i)}}$ **et** $\mathbf{r_{\tilde{\omega}} \sim U_{(r_{min};r_{max})}}$). Le nouvel objet devient $\tilde{\boldsymbol{\omega}}$ ($\mathbf{c_{\tilde{\omega}}}, \mathbf{r_\omega}$).

2. Remplacement de $\boldsymbol{\omega}$ par $\tilde{\boldsymbol{\omega}}$ dans la configuration $\mathbf{x^i}$, la nouvelle configuration devient $\tilde{\mathbf{x}}_{\mathbf{déplacement}} = \{\mathbf{x^i} \backslash \{\boldsymbol{\omega}\}\} \cup \{\tilde{\boldsymbol{\omega}}\}$.
3. Calcul du $\mathbf{GR}$ correspondant (Eq 11).

c- Acceptation/ refus de la nouvelle configuration : après la simulation d'un mouvement, l'algorithme décide d'accepter ou de refuser la nouvelle configuration :

1. $\boldsymbol{\alpha} = \mathbf{min}\ (\mathbf{1}, \mathbf{GR})$
2. Choisir uniformément une valeur aléatoire $\mathbf{accept}$ dans $[\mathbf{0}, \mathbf{1}]$ ($\mathbf{accept} \sim \mathbf{U}_{(\mathbf{0},\mathbf{1})}$).
3. Si $\mathbf{accept}$ est inférieure à $\boldsymbol{\alpha}$, le mouvement est accepté et $\mathbf{x^{i+1}} = \tilde{\mathbf{x}}$. Sinon le mouvement est refusé et $\mathbf{x^{i+1}} = \mathbf{x^i}$.

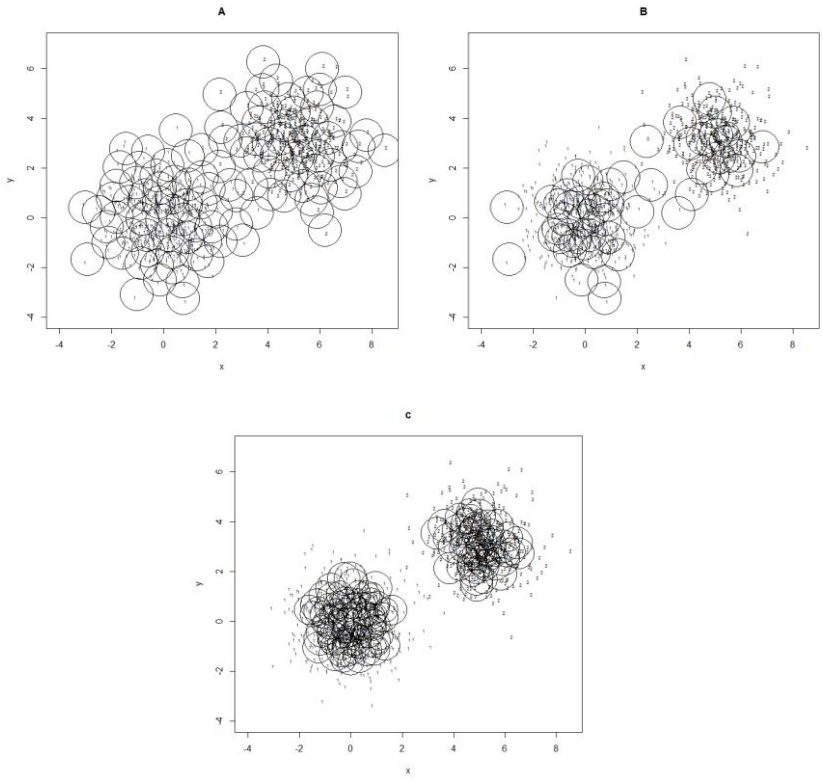

Figure 3. ***Différentes configurations du processus.***

À la fin de l'algorithme, les objets se localisent sur les régions à forte concentration de données (figure 3.b et 3.c). Les observations couvertes sont les observations prototypes détectées. Nous utilisons alors une version améliorée de l'algorithme KNN [6] pour l'affectation des observations non-prototypes. En effet, cette procédure d'affectation consiste à calculer la plus petite distance entre observations prototypes et non-prototypes, on classifie l'observation non-prototype la plus proche à la classe correspondante et on refait le même traitement.

3.2. Ajustement des paramètres du modèle

Le vecteur des paramètres du modèle est $\theta = (n_{min}; d_{min}; \beta; \gamma; r_{min}; r_{max})$

Les paramètres $n_{min} et\ d_{min}$ représentent respectivement le nombre de points minimal que doit couvrir un objet et la densité minimale de cet objet. Ils sont fixés respectivement à la moyenne de nombre de points que couvrent les objets de la configuration initiale et la moyenne de leurs densités.

Le paramètre d'intensité β représente le nombre moyen de points dans le cas d'un processus ponctuel de Poisson. Afin de favoriser la création d'objets selon la taille de l'ensemble d'observations, nous avons fixé sa valeur à $\log(n(y))$.

Si le paramètre β a un lien avec le nombre d'objets, le paramètre y est en rapport avec le nombre de composantes connexes. Si les données s'éparpillent dans l'espace, il serait préférable de favoriser la création des composantes connexes afin de couvrir un nombre important de régions à forte concentration. Nous proposons de définir γ par la variance de l'espace de données (σ^2), (la trace de la matrice de variances-covariances).

Les paramètres r_{min} et r_{max}représentent la borne de l'intervalle des valeurs du rayon d'un objet. Afin d'assurer que tout objet recouvre au moins deux observations, nous proposons d'attribuer au paramètre r_{min} la distance minimale entre toutes les observations. r_{max} est le rayon maximal que peut avoir un objet, c'est un paramètre très important, il influence le nombre de classes détectées. Une valeur importante ne permettra pas une bonne séparation des classes et au contraire une faible valeur donnera plusieurs partitions dans la même classe. r_{max} dépend de la taille des données, du degré de dispersion des données et de la dimension de l'espace. Nous avons choisi de nous inspirer de la méthode utilisée dans [5] et [15] pour fixer ce paramètre. Cette démarche consiste à passer par une phase d'apprentissage, afin de trouver la meilleure valeur de r_{max}, nous exécutons l'algorithme pour différentes valeurs de r_{max} et nous prenons la valeur médiane de l'intervalle qui assure une stabilité dans le nombre de modes détectés (Figure 4).

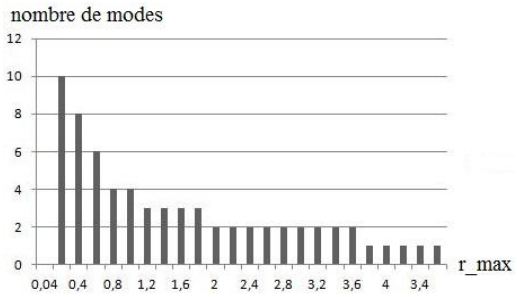

Figure 4. ***Nombre de modes détectés selon le rayon r_max.***

4. Résultats expérimentaux

Nous avons appliqué l'algorithme proposé sur des données multidimensionnelles de différents types que nous avons générées : séparables linéairement ; non séparables linéairement et enchevêtrées. Ensuite, nous avons testé, pour détecter les différentes classes, ensuite nous avons testé l'algorithme sur des données réelles issues des données des puces à ADN. Nous avons comparé l'algorithme MDMPP-KNN proposé avec K-means [1], Mean-shift [16], Classification spectrale [17] et l'algorithme basé sur les champs de Markov [5].

4.1. Données simulées :

Le premier ensemble de données que nous avons utilisées est composé de trois classes générées suivant une distribution Gaussienne et comportant 400 observations chacune (Figure 5.a). L'algorithme s'est stabilisé après l'itération 10000. La Figure 5.b montre la localisation des objets dans les régions à forte concentration de données.

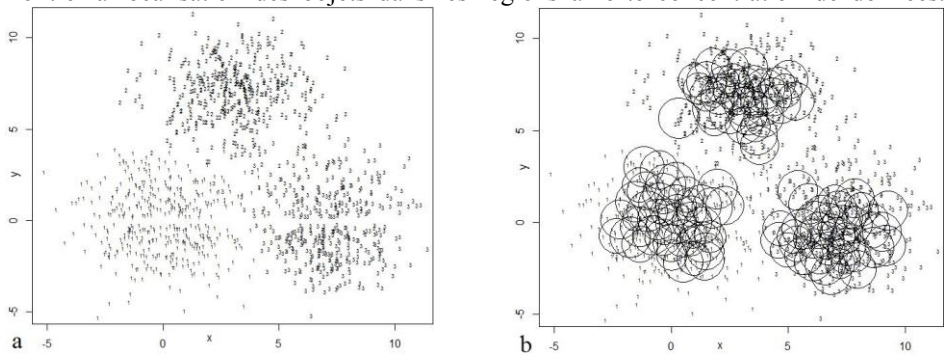

Figure 5. ***Données à séparation linéaires : a. données brutes. b. Prototypes des classes détectés.***

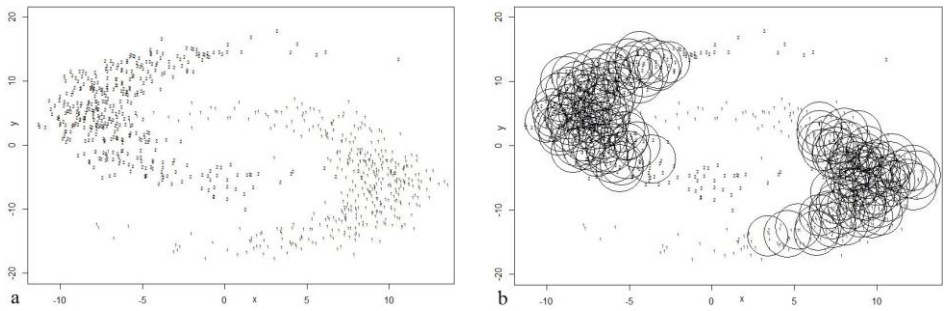

Figure 6. ***Données à séparation non linéaires : a. données brutes. b. Prototypes des classes détectés.***

Les figures 8 et 9 montrent l'évolution du nombre d'objets et de l'énergie du processus dans les trois exemples. Nous remarquons que ces deux termes convergent.

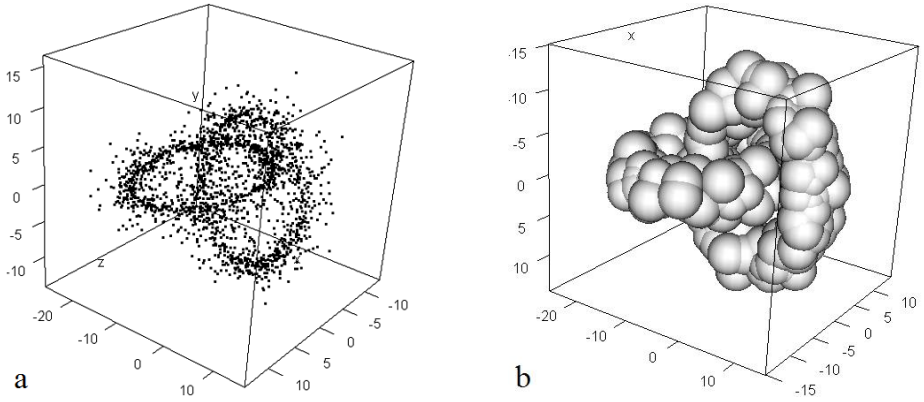

Figure 7. ***Données enchevêtrées multidimensionnelles : a. données brutes. b. Prototypes des classes détectés.***

Les résultats de classification sont donnés dans le Tableau 1. La matrice de confusion montre les observations mal classées. Une comparaison entre les taux d'erreur des différents algorithmes prouve l'efficacité de notre algorithme. Nous remarquons que les algorithmes ont donné des taux d'erreurs comparables sur les données linéairement séparables, mais dans le cas des données plus complexes, notre algorithme et l'algorithme de [5] ont donné des taux d'erreurs relativement faibles par rapport à l'algorithme k-means.

		Exemple 1	Exemple 2	Exemple 3
Notre Algorithme	Matrice de confusion	$\begin{pmatrix} 387 & 4 & 2 \\ 3 & 390 & 2 \\ 10 & 6 & 396 \end{pmatrix}$	$\begin{pmatrix} 390 & 13 \\ 10 & 387 \end{pmatrix}$	$\begin{pmatrix} 2000 & 1 \\ 0 & 1999 \end{pmatrix}$
	Taux d'erreur	2.25%	2.87%	0.025%
Algorithme de [5]	Matrice de confusion	$\begin{pmatrix} 388 & 4 & 6 \\ 2 & 394 & 3 \\ 10 & 2 & 391 \end{pmatrix}$	$\begin{pmatrix} 388 & 14 \\ 12 & 386 \end{pmatrix}$	$\begin{pmatrix} 1999 & 2 \\ 2 & 1997 \end{pmatrix}$
	Taux d'erreur	2.25%	3.25%	0.05%
K-means	Matrice de confusion	$\begin{pmatrix} 391 & 7 & 10 \\ 3 & 389 & 3 \\ 6 & 4 & 387 \end{pmatrix}$	$\begin{pmatrix} 363 & 36 \\ 37 & 364 \end{pmatrix}$	$\begin{pmatrix} 1284 & 642 \\ 716 & 1358 \end{pmatrix}$
	Taux d'erreur	2.75%	9.125%	33.95%
Mean-shift	Matrice de confusion	$\begin{pmatrix} 393 & 7 & 11 \\ 1 & 389 & 5 \\ 6 & 4 & 384 \end{pmatrix}$	$\begin{pmatrix} 367 & 22 \\ 33 & 378 \end{pmatrix}$	$\begin{pmatrix} 2000 & 1735 \\ 0 & 265 \end{pmatrix}$
	Taux d'erreur	2.83%	6.825%	43.375%
Classification spectrale	Matrice de confusion	$\begin{pmatrix} 387 & 4 & 2 \\ 3 & 390 & 2 \\ 10 & 6 & 396 \end{pmatrix}$	$\begin{pmatrix} 389 & 8 \\ 11 & 392 \end{pmatrix}$	$\begin{pmatrix} 1994 & 353 \\ 6 & 1647 \end{pmatrix}$
	Taux d'erreur	2.25%	2.375%	8.975%

Tableau 1. ***Tableau comparatif entre notre algorithme, algorithme de [5], k-means, Mean-shift et la classification spectrale.***

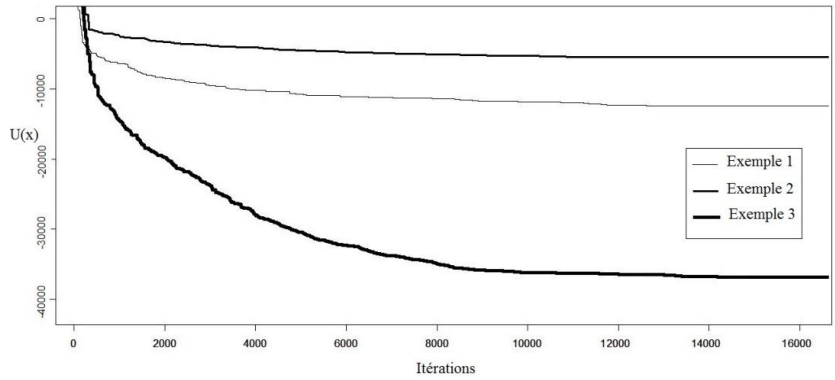

Figure 8. ***Évolution de l'énergie du processus (Exemple 1 : bidimensionnelle figure 5, Exemple 2 : bidimensionnelle non linéaire figure 6 et Exemple 3 : multidimensionnelle enchevêtrées figure 7).***

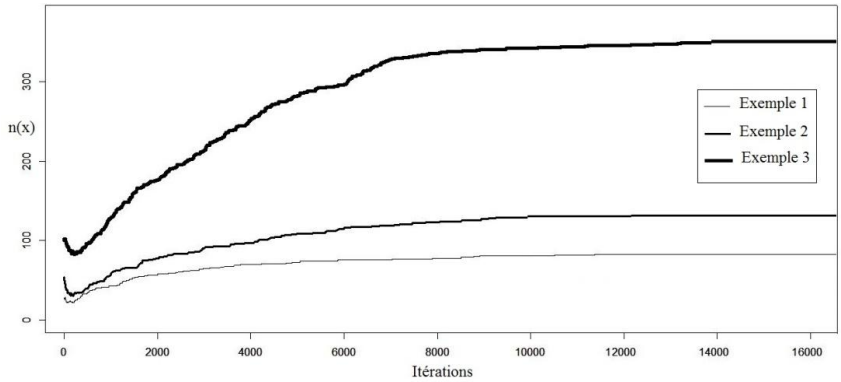

Figure 9. ***Évolution du nombre d'objets dans le processus (Exemple 1 : bidimensionnelle figure 5, Exemple 2 : bidimensionnelle non linéaire figure 6 et Exemple 3 : multidimensionnelle enchevêtrées figure 7).***

Les figures 8 et 9 montrent la stabilisation du nombre d'objets et de l'énergie du processus pour chaque ensemble de données.

4.2. Données réelles :

Nous nous sommes intéressées aux données génomiques issues des puces à ADN [18]. Nous avons appliqué notre algorithme MDMPP-KNN sur deux ensembles de données de références :

Leucémie : les données sont organisées sous la forme de matrice numérique, où les lignes représentent les gènes et les colonnes sont les expressions des gènes. La matrice est de 38 lignes et 100 colonnes avec 3 classes connues.

Données de plusieurs tissus: l'ensemble de données est constitué de 103 tissus et de 1000 gènes. Les tissus se regroupent en 4 classes.

L'histogramme présenté dans la figure 10 montre les taux d'erreurs des algorithmes sur les deux ensembles de données. Il montre que Mean-shift et la classification spectrale ont donné des taux d'erreur très élevés sur les données réelles et que MDMPP-KNN et k-means ont donné des taux d'erreur relativement faibles par rapport aux autres algorithmes, ceci prouve l'efficacité de MDMPP-KNN sur des données réelles.

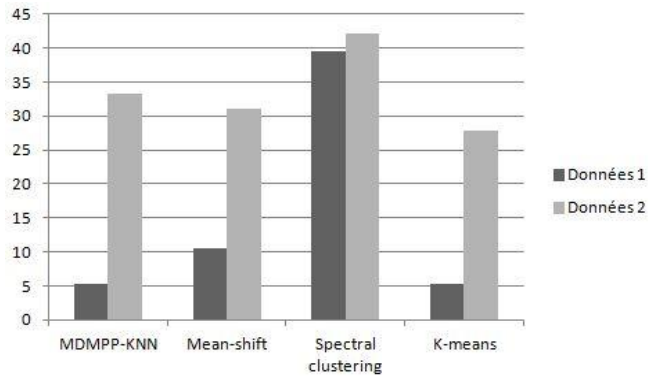

Figure 10. ***Comparaison du taux d'erreur pour les deux ensembles de données.***

5. Conclusion

Dans ce papier, nous avons proposé l'algorithme MDMPP-KNN qui est un nouvel algorithme de classification automatique de données multidimensionnelles. Il est le résultat de la succession des méthodes MDMPP et KNN. MDMPP est un algorithme de détection des prototypes de classes; il exploite le formalisme des processus ponctuels marqués. KNN est l'algorithme qui affecte les observations non prototypes. La fonction de densité de probabilité du processus ponctuel marqué est proportionnelle à l'exponentielle de l'énergie de Gibbs. La modélisation proposée pour la fonction d'énergie du processus permet de favoriser l'acceptation des objets bien positionnés. La configuration finale du processus est composée d'objets interconnectés localisés sur les régions à forte concentration de données. Les valeurs des paramètres de la méthode seront estimées à partir de la distribution de la configuration initiale. Les résultats de la détection des prototypes des classes ont montré l'efficacité de l'algorithme proposé. Nous avons comparé notre méthode avec différentes méthodes telles que K-means, Mean-shift, la classification spectrale et une méthode développée dans [5]. MDMPP-KNN a donné des résultats de classification relativement meilleurs. En perspective à ces travaux, nous souhaitons améliorer cet algorithme en introduisant l'information ontologique afin de permettre une meilleure classification des données biologiques.

6. Bibliographie

[1] V. FABER, "Clustering and the continuous k-means algorithm", *Los Alamos Science*, vol 1(22), page 138-144, 1994.

[2] C. BIERNACKI, G. CELEUX, et G. GOVAERT. "Assessing a mixture model for clustering with the integrated completed likelihood". *IEEE Trans. on PAMI*, vol 22, page 719-725, 2000.
[3] M. Ester, H. P. Kriegel, J. Sander et X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, *in Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, page 226–231, 1996.
[4] J. Sander, M. Ester, H.P. Kriegel and X. Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, vol 2(2), page 169-194, 1998.
[5] A. MOUSSA, A. SBIHI et J.-G. POSTAIRE, "A Markov random field model for mode detection in cluster analysis", *Science Direct, Pattern Recognition Letters*, vol 29, page 1197-1207, 2008.
[6] S. OUGIAROGLOU et al. "Adaptive *k*-Nearest-Neighbor Classification Using a Dynamic Number of Nearest Neighbors", *Advances in Databases and Information Systems*, vol 46(90), page 66-82, 2007.
[7] P. CLIFFORD, "Markov Random Fields in statistics", *Disorder in Physical Systems: A Volume in ~Honour of John M. Hammersley*, pages 19-32, 1990.
[8] R.S. STOICA, E. GAY et A. KRETZSCHMAR, "Cluster pattern detection in special data based on Monte Carlo Inference", *Biometrical Journal*, vol 49(4), page 505-519, 2007.
[9] R.S. STOICA, V.J. MARTINEZ et E. SAAR, "Filaments in observed and mock galaxy catalogues", *Astronomy and Astrophysics*, vol 510 (A38), page 1-12, 2010.
[10] O. ALATA, S. BURG et A. DUPAS, "Grouping/degroupin point process, a point process driven by geometrical an topological properties of a partition in regions", *Computer Vision and Image Understanding*, vol 115(9), page 1324-1339, 2011.
[11] M.S. KULIKOVA, I.H. JERMYN,X. DESCOMBES, E. ZHIZHINA et J. ZERUBIA, "A Marked Point Process Model Including Strong Prior Shape Information Applied to Multiple Object Extraction From Images", *International Journal of Computer Vision and Image Processing*, vol 1(2),page 1-12, 2011.
[12] Y. C. CHIN et A. J. BADDELEY, On connected component markov point processes, Advances in Applied Probability, vol 31, page 279–282, 1999.
[13] Y. C. CHIN et A. J. BADDELEY, Markov interacting component processes, Advances in Applied Probability, vol 32(3), page 597–619, 2000.
[14]: P. J. GREEN, "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination ", *Biometrika*, vol 82, page 711–732, 1995.
[15] J. G. POSTAIRE et C. P. C. VASSEUR, An approximate solution to normal mixture identification with application to unsupervised pattern classification, *IEEE Trans. Pattern Anal. Machine Intell*, vol 3(2), page 163–179, 1981.
[16] X. T. YUAN, Agglomerative mean-shift clustering, *IEEE Transactions on Knowledge and Data Engineering*, vol 24, page 209–219, 2012.
[17] C. ARIAS, G. CHEN et G. LERMAN, Spectral clustering based on local linear approximations, *Electronic Journal of Statistics*, vol 5, page, 1537–1587, 2011.

[18] R. GIANCARLO, L. BOSCOL, G, L. PINELLO et F. UTRO, A methodology to assess the intrinsic discriminative ability of a distance function and its interplay with clustering algorithms for microarray data analysis, *BMC Bioinformatics*, vol 14 (S-1), page S6, 2013.