Classification paramétrique robuste partiellement supervisée en reconnaissance des formes

Christophe Saint-Jean

Université de La Rochelle - UFR Sciences
Laboratoire d'Informatique et d'Imagerie Industrielle

Année :

Numéro attribué par la bibliothèque

THÈSE

pour l'obtention du grade de

DOCTEUR DE l'UNIVERSITÉ DE LA ROCHELLE

(spécialité Informatique)

présentée et soutenue publiquement le 17 Décembre 2001

par

Christophe SAINT-JEAN

Classification paramétrique robuste partiellement supervisée en reconnaissance des formes

Directeur de thèse : Carl FRÉLICOT

Composition du jury

Président :	Christian OLIVIER	Université de Poitiers
Rapporteurs :	Hubert EMPTOZ	Institut National des Sciences Appliquées de Lyon
	Jack-Gérard POSTAIRE	Université de Lille I
Examinateurs :	Bertrand VACHON	Université de La Rochelle
	Carl FRÉLICOT	Université de La Rochelle

Je dédie cette thèse à ma famille et à mes amis qui m'ont soutenu tout au long de ce parcours.

Remerciements

Je tiens en premier lieu à remercier Mr Carl Frélicot, Maître de Conférences à l'université de La Rochelle, d'avoir proposé et encadré ce sujet pendant trois ans. Je lui exprime ma profonde gratitude pour m'avoir fait profiter de ses connaissances, mais aussi de ses méthodes de travail, et surtout de sa rigueur scientifique. Grâce à lui, j'ai découvert un domaine de recherche qui aujourd'hui me passionne.

J'adresse mes plus vifs remerciements à Mr Hubert Empotz, Professeur à l'Institut National des Sciences Appliquées de Lyon, et à Mr Jack-Gérard Postaire, Professeur à l'Université de Lille I, pour avoir accepté d'être les rapporteurs de ce mémoire, pour la pertinence de leurs remarques, et leur participation à ce jury.

Je remercie également Mr Christian Olivier, Professeur à l'Université de Poitiers, d'avoir accepter de la présidence de ce jury.

J'adresse mes plus vifs remerciements à Mr Bertrand Vachon, Professeur à l'Université de La Rochelle, pour sa participation à ce jury de thèse. Il a été l'initiateur de ce projet et m'a prodigué de précieux conseils concernant la gestion d'une thèse.

En parallèle de cette thèse, j'ai découvert le plaisir (et les difficultés) de l'enseignement. Je suis ainsi reconnaissant aux divers responsables d'unités d'enseignement de m'avoir confié une infime partie de l'élévation de la nation.

Enfin, je souhaite témoigner toute mon amitié à l'ensemble de mes collègues thésards du L3i pour leur soutien technique, logistique, moral à cette thèse et surtout pour tous les bons moments passés ensemble. Je ne saurai oublier les doctorants d'autres disciplines que j'ai eu le bonheur de côtoyer dans le cadre de la vie associative.

Table des matières

Table des figures

Liste des tableaux

Notations

p	taille du vecteur forme
m	indice relatif à un numéro d'attribut $m = 1, \ldots, p$
χ	ensemble de données pour l'apprentissage, $Card(\chi) = n$
n	nombre de points dans l'ensemble d'apprentissage
i, j	indices relatifs à un numéro d'élément $i, j = 1, \ldots, n$
x_i	i-ième élément de l'ensemble d'apprentissage
x_i^m	valeur du m-ième attribut de x_i
c	nombre de classes
k, l	indices relatifs à un numéro de classe $k, l = 1, \ldots, c$
π_k	probabilité a priori de la classe k
μ_k	vecteur moyenne de la classe k
Σ_k	Matrice de covariance de la classe k
χ^u	ensemble des données d'apprentissage non supervisées $Card(\chi^u) = n_u$
χ^d	ensemble des données d'apprentissage supervisées $Card(\chi^d) = n_d$
χ_k^d	ensemble des données d'apprentissage supervisées appartenant à la classe k
n_u	nombre de points non supervisés
n_d	nombre de points supervisés
$x_{i,k}$	i-ième élément supervisé appartenant à la classe k
$f(x_i; \Theta_k)$	densité a priori de x_i conditionnellement à la classe k paramètrée par Θ_k

Acronymes

ACP	Analyse en Composantes Principales	(1.1.2.3)
AFD	Analyse Factorielle Discriminante	(1.1.2.3)
AFCE	Adaptive Fuzzy C-Elliptotypes	(2.3.1.1.3)
AFCS	Adaptive Fuzzy C-Shells	(2.3.1.1.3)
AIC	Akaïke Information Criterion	(2.4.1)
BIC	Bayesian Information Criterion	(2.4.1)
CEM	Classification EM	(2.2.4.2)
CWEM	Component-Wise EM	(2.2.4.3)
ECD	Extraction de Connaissances dans des bases de Données	(2)
EM	Expectation-Maximization	(2.2.3)
FCC	Fuzzy C-Covariance	(2.3.1.1.2)
FCM	Fuzzy C-Means	(2.3.1.1)
FCL	Fuzzy C-Lines	(2.3.1.1.3)
FCE	Fuzzy C-Elliptotypes	(2.3.1.1.3)
FCV	Fuzzy C-Varieties	(2.3.1.1.3)
FCS	Fuzzy C-Shells	(2.3.1.1.3)
FMLE	Fuzzy Maximum Likelihood Estimation	(2.3.1.2)
FNC	Fuzzy Noise Clustering	(3.1.3.3)
HCM	Hard C-Means	(2.3.1.1)
ICOMP	Information COMPlexity	(2.4.1)
JPEG	Joint Photographic Experts Group	(2.4.5)
MCEM	Monté-Carlo EM	(2.2.4.4)
MP3	MPeg 1-layer 3	(2.4.5)
NEC	Normalized Entropy Criterion	(2.4.1)
NEM	Neighborhood EM	(2.2.4.4)
PCM	Possibilistic C-Means	(2.3.1.1)
RCA	Robust Competitive Algorithm	(3.1.3.2)
RdF	Reconnaissance des Formes	(1.1)
SAEM	Simulated annealing EM	(2.2.4.4)
SEM	Stochastic EM	(2.2.4.1)
SMEM	Split-Merge EM	(2.2.4.4)

Chapitre 1

Introduction

Cette partie présente de manière synthétique la problématique de la reconnaissance des formes. Notre propos, illustré par des exemples, décrit le fonctionnement global d'un dispositif de reconnaissance des formes. Cette présentation se veut à la fois didactique mais non exhaustive tant le sujet est vaste. Ces préliminaires permettront dans un second temps de préciser le cadre général et les objectifs de cette étude.

1.1 Généralités sur la Reconnaissance des Formes

La Reconnaissance des Formes (RdF) est une discipline entre les mathématiques et l'informatique définie par divers auteurs comme :

> "Pattern recognition, a field concerned with machine recognition of meaningful regularities in noisy and complex environments" (Duda et Hart [DH73])
> "pattern recognition is a search for structure in data." (Bezdek [Bez87])

Ces deux définitions laissent apparaître à la fois l'objectif et la difficulté de la tâche à accomplir. La RdF, ainsi définie, est l'apprentissage ou la découverte de structures appelées classes dans un ensemble de données éventuellement perturbées.

L'objectif de cet apprentissage est généralement décisionnel suivant qu'il s'agit de déclencher un événement, une action en fonction de l'appartenance de tel ou tel objet à une classe. Suivant le domaine d'application visé, l'information traitée est de nature qualitative, quantitative ou des deux à la fois. Dans cette section, l'intérêt de la reconnaissance des formes est illustré à travers quelques exemples d'application. Par la suite, les diverses étapes d'un système générique seront présentées de manière synthétique.

1.1.1 Exemples d'application

La plupart des applications décrites ici sont apparues au cours de ces vingt dernières années en même temps que la formidable augmentation de la puissance des ordinateurs.

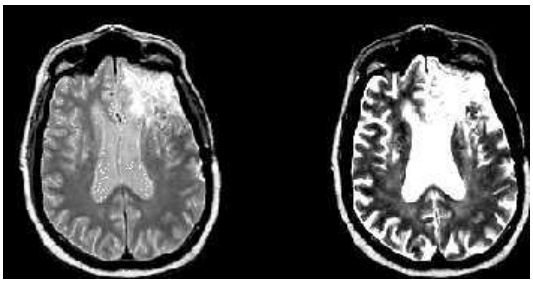


FIG. 1.1 – Aide au diagnostic médical : Segmentation d'une image IRM

Cependant, la reconnaissance de formes en tant que discipline scientifique est antérieure aux années 60, avec des travaux théoriques principalement dans le domaine des statistiques [Fis36]. Seul l'avènement de l'informatique a permis d'appliquer ces techniques aux traitement de volumes de données conséquents.

1.1.1.1 La vision artificielle

La vision par ordinateur est sans doute l'un des principaux champs d'application de la Reconnaissance des Formes (RdF). De nos jours, elle a trouvé une place prépondérante dans un domaine tel que l'imagerie bio-médicale. Du diagnostic à la chirurgie assistée par ordinateur en passant par la simulation, les techniques d'imagerie constituent une aide constante au praticien. Par exemple, dans la détection des cellules cancéreuses du cerveau, les images sont acquises en Imagerie par Résonance Magnétique (IRM)[LFHM01]. Au cours de l'étape de segmentation, on va rechercher dans les images différentes régions du cerveau : matière blanche, matière grise, liquide cérébro-spinal, etc ... (voir Fig. (1.1)). On cherche ensuite à détecter d'éventuelles tumeurs bénignes ou malignes à partir de caractéristiques extraites de régions de l'image. Bien évidemment, il ne s'agit là que de fournir une aide au diagnostic validée ultérieurement par le praticien tant le coût associé à un mauvais diagnostic est grand.

En biologie marine, certaines bactéries prennent une forme hexagonale en présence de toxines particulières. En couplant un microscope avec un ordinateur, on peut mettre en place une méthode de recherche de motifs hexagonaux dans une image de ces bactéries. Des systèmes fondés sur ce principe ont déjà été mis en oeuvre pour déceler (indirectement) des traces de pollution.

Dans un autre domaine, les industriels cherchent à contrôler la qualité de fabrication de leurs produits. La vision artificielle peut être utilisée dans un système de détection de défauts [NJ95]. On peut imaginer diverses pièces usinées passant sur un tapis roulant et filmées en continu par une caméra. Avec un traitement en temps réel de cette image, on

peut, par exemple, vérifier si l'angle formé par deux pièces assemblées est conforme au cahier des charges.

1.1.1.2 La reconnaissance des caractères et de mots

La reconnaissance des caractères est une application classique de la RdF [GS90]. Elle est utilisée aussi bien par les particuliers que par le secteur privé. Elle vise à transformer un texte manuscrit ou typographié, sur tout type de support, vers un texte "numérique" (éditable par un traitement de texte). Le document, sous forme papier par exemple, est préalablement numérisé par un scanner pour en obtenir une représentation sous la forme d'une image. On essaie ensuite d'extraire chacune des lettres présentes sur l'image et d'en déterminer un équivalent "informatique" (voir Fig. (1.2)). Dans les applications actuelles les plus sophistiquées, on enrichit le système d'informations supplémentaires sur le texte à reconnaître en utilisant des grammaires contextuelles et autres règles linguistiques [SRI99]. Aujourd'hui, ce type d'application est couramment utilisé à des fins d'archivage, d'indexation, tri automatique du courrier, traitement des chèques, etc...

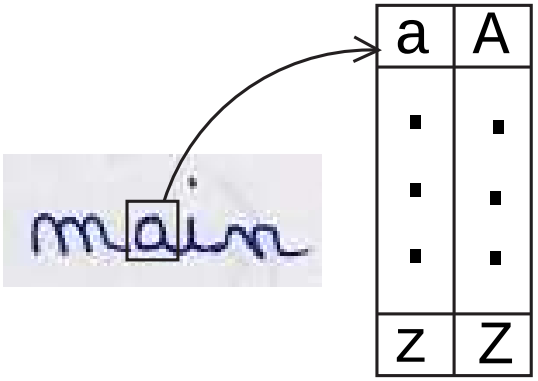


FIG. 1.2 – Reconnaissance des caractères : Reconnaître "a" dans la fenêtre

1.1.1.3 Quelques autres applications

La reconnaissance vocale est une utilisation relativement récente des techniques de la RdF donnant lieu à de nombreuses recherches [CMU+98]. D'une manière générale, l'objectif est de faciliter la communication humaine et Homme/Machine en reconnaissant l'information parlée. La traduction automatique de la parole ou la dictée vocale en sont des applications potentielles au même titre de la commande vocale. Dans ce dernier cas, l'instruction parlée est envoyée deux fois plus rapidement que si elle était dactylographiée. La commande vocale peut également permettre à des personnes handicapées de contrôler des machines en leur parlant.

Les applications présentées jusqu'ici sont essentiellement en rapport des informations sensorielles (visuelles, auditives). Ce n'est pas là une contrainte de la RdF qui peut également s'appliquer pour tout autre problème mettant en jeu une analyse des données. On peut citer les applications suivantes :

- L'analyse des fins de parties du jeu d'échecs (La partie est elle finie ou non ?),
- attribution de prêts bancaires,
- détection de modes de fonctionnement (transitoire, nominal, défaut) d'un processus complexe pour le diagnostic
- la reconstruction de l'arbre phylogénétique[1]
- détection de fraudes dans les réseaux téléphoniques,
- l'indexation de pages WEB à partir de certains mots-clés fréquents,
- détection des élèves en avance ou en retard par le résultat de test psychologiques et psycho-moteurs.
- etc..

Les applications décrites ici ne sont que quelques-unes parmi d'innombrables. Le principe général de tels systèmes va maintenant être décrit plus en détails.

1.1.2 Le schéma de la reconnaissance des formes

L'objectif de la reconnaissance des formes est de classifier des entités en catégories à partir d'observations effectuées sur celles-ci. Ce dispositif se décompose généralement en 5 étapes (voir figure 1.3) :

- Acquisition des données ;
- Génération de caractéristiques ;
- Extraction/Sélection des caractéristiques ;
- Classification ;
- Évaluation du système.

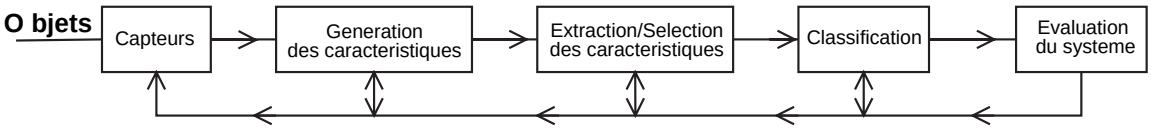


FIG. 1.3 – Le processus de la reconnaissance de formes

De l'acquisition des données à la classification, la qualité du système dépend grandement des étapes précédentes. Pour illustrer le déroulement standard de ce processus, prenons l'exemple d'un système d'aide à la conduite automobile dont l'objectif est de prévenir le conducteur d'un éventuel danger par l'intermédiaire de divers capteurs.

[1] Elaboration d'une hiérarchie entre les espèces basée sur une mesure de distance entre les codes ADN

1.1.2.1 Acquisition des données

Dans une application, les données acquises peuvent être de type et de nature différentes et dépendent de ce que l'on recherche. Dans le cas de notre exemple, on peut avoir :

- des informations sur l'état du véhicule telles que la position, la vitesse, l'accélération, pression de freinage, etc ... (capteurs proprioceptifs) ;
- des informations sur l'environnement du véhicule avec le GPS, la télémétrie, des caméras, etc... (capteurs extéroceptifs) ;
- des informations propres au véhicule comme le modèle, la couleur, la plaque minéralogique, etc... ;
- des informations sur le conducteur (hypovigilance, âge, taille, sexe).

Dans cette étape, la pertinence des données acquises vis-à-vis de l'application n'est pas étudiée (comme la couleur du véhicule). Par contre, il est nécessaire de prendre en compte l'incertitude liée aux différents capteurs. On sait par exemple que les systèmes GPS standard commettent une erreur moyenne de 10 mètres.

Certains capteurs peuvent également se révéler défaillants et retourner des valeurs erronées voire aberrantes. Par la suite, ces données seront également appelées "outliers" (terme anglophone consacré). Ces points peuvent perturber toutes les phases du processus de reconnaissance en introduisant un biais dans les différents estimateurs. Il est important qu'un système robuste traite correctement ce type de données.

Certains problèmes sont inhérents au type de données acquises en raison de la taille qu'elles nécessitent pour le stockage. C'est le cas par exemple pour toutes les informations de type audiovisuelles. En effet, la faible capacité (relative) des machines actuelles nécessite de les comprimer, le plus souvent avec perte. Cette perte d'information peut ainsi faire apparaître des problèmes spécifiques liés à la technique de compression (Ex. : artefacts "carrés" du JPEG). Là encore, il convient donc d'être attentif sur le mode de stockage des données qui peut introduire une perturbation de celles-ci.

Une fois les données acquises, on possède une collection d'objets hétérogènes. On peut dès lors passer à la phase de génération des caractéristiques.

1.1.2.2 Génération des caractéristiques

L'étape de génération des caractéristiques est primordiale dans le succès de la reconnaissance. Celle-ci est généralement effectuée en collaboration avec un expert du domaine traité. Parmi les données acquises, certaines seront utilisées directement (Ex. : la vitesse, la position du véhicule), d'autres seront transformées (Ex. : la vidéo, les images), d'autres encore seront simplement éliminées car considérées comme non pertinentes par l'expert

(Ex. : la couleur, la plaque minéralogique).

La transformation des données est particulièrement complexe dans le cas des images et de la vidéo. Dans notre exemple, les images peuvent servir à déterminer la position du véhicule sur la chaussée en détectant le marquage au sol, d'éventuels obstacles, la luminosité extérieure, etc... On retrouve alors les problèmes traditionnels de l'analyse d'images tels que le filtrage, la segmentation, la détection de lignes. Le lecteur intéressé par ces aspects pourra se reporter à un ouvrage de référence [CP95].

Certains ouvrages et articles recensent un catalogue des caractéristiques traditionnellement extraites dans les domaines de l'analyse d'images et du signal sonore [TK99] [TJT96] [Lon98].

Les attributs peuvent s'écrire sous la forme matricielle suivante :

$\chi = \{x_i\} =$

	Position	Vitesse	...	*Caractéristique m*	...	âge	sexe
Entité 1	0.25	80		x_1^m		52	H
⋮	⋮	⋮		⋮		⋮	⋮
Entité i	0.22	110	...	x_i^m	...	29	F
⋮	⋮	⋮		⋮		⋮	⋮
Entité n	0.24	140	...	x_n^m	...	19	H

où les $x_i = [x_i^1 \dots x_i^p]^T$ sont également appelés les vecteurs-forme. Les variables n et p désignent respectivement le nombre d'entités et le nombre d'attributs par entité. Dans la suite et pour tout le manuscrit, la même désignation sera appliquée

Il est parfois nécessaire, et lorsque cela est possible, de retourner à l'étape d'acquisition lorsque les données acquises sont de mauvaise qualité : valeurs erronées, résolution trop faible, représentativité des données (biais de sélection), conditions d'acquisition (Ex. : conditions d'éclairage), valeurs manquantes.

Lorsque l'étape de génération est terminée, le nombre de caractéristiques peut être important. On convient donc de faire une sélection parmi celles-ci et/ou d'en extraire de nouvelles par combinaison. Ceci est le rôle de la phase d'extraction/sélection de caractéristiques.

1.1.2.3 Extraction/Sélection des caractéristiques

L'extraction/sélection des caractéristiques permet de réduire la dimensionnalité des données. Nous noterons Ψ l'ensemble des attributs issus de la phase de génération avec

$Card(\Psi) = p$.

Les méthodes de réduction de dimension sont nombreuses et ont pour objectif de conserver le maximum d'information possible dans un espace de dimension inférieure. On fera ici la distinction entre les méthodes d'extraction qui créent de nouvelles variables à partir des anciennes et les méthodes de sélection qui cherchent seulement un sous-ensemble d'attributs optimal suivant un critère donné. Pour justifier la réduction de la dimension, le lecteur peut se reporter à la section 2.4.3 du second chapitre.

Cette notion de quantité d'information est relative suivant la méthode utilisée. Parmi les méthodes d'extraction les plus employées, citons :

- **L'analyse en composantes principales (ACP)** où l'on suppose que l'information est principalement portée par les axes de grande variance. On aboutit à définir un sous-espace de dimension p' ($p' \leq p$) engendré par les vecteurs propres de la matrice de covariance correspondants aux p' plus grandes valeurs propres. L'ACP maximise la variance expliquée [Sap90].
- **L'analyse factorielle discriminante (AFD)** où l'on cherche le sous-espace vectoriel pour lequel l'inertie intra-classes est minimale (classes compactes) et l'inertie inter-classes est maximale (classes séparées). Cette méthode est supervisée car elle suppose la connaissance de la classe d'appartenance pour un certain nombre d'entités.
- **La projection de Sammon** est une technique de projection non linéaire particulièrement adaptée à la visualisation des données en 2 ou 3 dimensions [Sam69]. L'idée est de respecter au mieux la topologie en préservant les distances entre les points dans l'espace de dimension inférieure. On cherche à minimiser l'erreur :

$$E = \frac{1}{\sum_{j>i}^{n} D_p(x_i, x_j)} \sum_{j>i}^{n} \frac{(D_p(x_i, x_j) - D_{p'}(x'_i, x'_j))^2}{D_p(x_i, x_j)}$$

  où p' représente la dimension de l'espace dans lequel les x_i, x_j sont projetés en x'_i, x'_j. Comme cette technique réside dans la résolution d'un système non linéaire, la projection obtenue n'est pas unique.

Dans les deux premiers cas, de nouvelles variables sont crées par combinaisons linéaires des variables existantes. De telles méthodes sont également utilisées dans le contexte de l'extraction automatique des caractéristiques. Dans ce contexte, on acquiert un nombre maximal d'attributs que l'on réduit pour obtenir un espace de dimension inférieure dans lequel la discrimination est maximale (Ex. : AFD).

Les méthodes de sélection réduisent également la dimension en recherchant le sous-ensemble Ψ' de p' attributs parmi p qui maximise certain critère $J(\Psi')$. En mode supervisé, il semble naturel de chercher à maximiser la probabilité de classification correcte $1 - Prob(Erreur)$. Ce critère rend la méthode dépendante du discriminateur ou de la

règle de classement utilisés.

La recherche exhaustive du meilleur sous-ensemble possible selon un critère donné J est combinatoirement prohibitive ($C_p^{p'}$). Cependant, si le critère est monotone ($P'_1 \subset P'_2 \Longrightarrow J(P'_1) < J(P'_2)$), on peut utiliser une méthode non exhaustive de type Branch-and-Bound garantissant un optimum global [Bob88]. Malheureusement, la plupart des critères utilisés en pratique sont rarement monotones [JDM00]. On a donc recours à des stratégies sous-optimales comme les Sequential Forward Selection (SFS), Sequential Backward Selection (SBS) et Sequential Forward Floating Search (SFFS) [JDM00].

Lorsque l'acquisition, la génération et l'extraction/sélection des attributs est achevée, il reste à effectuer l'apprentissage proprement dit, la classification.

1.1.2.4 Classification

Le type d'une méthode de classification se décline généralement en 2 familles : le mode supervisé, le mode non supervisé. Si l'on dispose d'un ensemble de points étiquetés, on parlera de classification supervisée. Dans le cas contraire, on effectue une classification non supervisée appelée également classification automatique. Dans ce travail, on étudiera par la suite le cas d'un étiquetage partiel des données correspondant à un mode semi-supervisé.

On distingue deux approches duales :
- La recherche de frontières et de fonctions discriminantes.
- La recherche d'une partition des données en sous-ensembles appelées classes.

Afin de visualiser la problématique de la classification, les exemples suivants seront restreints au cas bi-dimensionnel.

En classification automatique, la recherche d'une partition des données revient à regrouper celles-ci selon une certaine mesure de similarité ou de dissimilarité (voir figure 1.4). La métrique employée peut être de nature géométrique ou probabiliste. On distinguera ainsi les méthodes paramétriques qui modélisent les données par un ensemble de paramètres à estimer, des méthodes non paramétriques qui ne font pas d'hypothèses sur la distribution de ces données. La cas des méthodes semi-paramétriques n'est pas abordée dans cette étude. Le chapitre 2 dénombre quelques approches possibles de la classification automatique.

Dans le cas supervisé, les classes d'appartenance des données sont connues. La recherche des frontières entre les classes peut être effectuée par la recherche des fonctions discriminantes (voir AFD page 7). Les Machines à Vecteurs de Support (MVS) recherchent

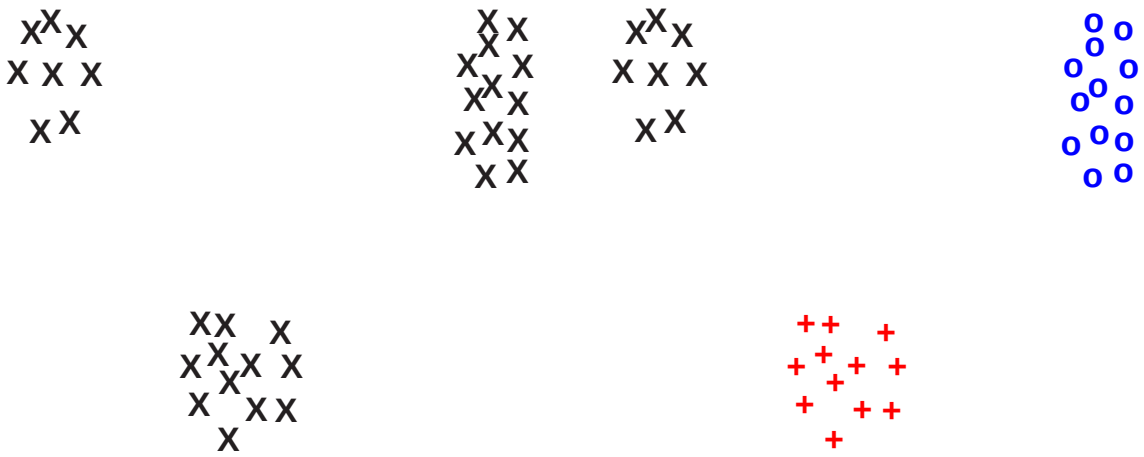


FIG. 1.4 – Classification par partition - Regroupement par similarité

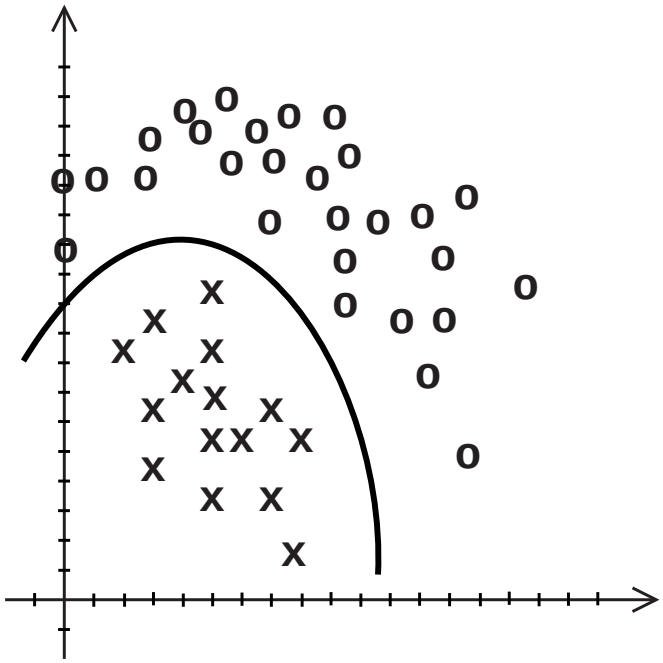


FIG. 1.5 – Détermination de la frontière optimale entre 2 classes

les frontières de décision maximisant la marge séparatrice entre les différentes classes (voir section 3.17). La figure (1.5) montre un exemple à deux classes séparables de manière optimale.

1.1.2.5 Évaluation du système

Une fois l'apprentissage effectué, le système doit être validé par une phase de test. On distinguera ici le classement de la classification précédemment décrite. Le test d'une

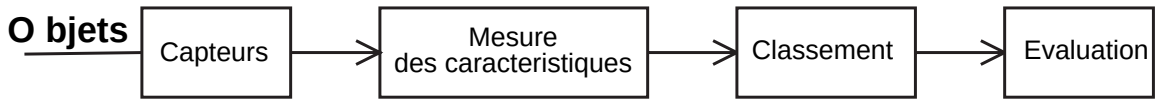


FIG. 1.6 – Les différentes étapes du test d'une donnée

donnée se déroule en trois étapes (voir Fig. (1.6)) :

- Acquisition des observations de l'objet à tester ;
- Extraction des mêmes caractéristiques utilisées lors de l'apprentissage ;
- Classement parmi les classes $C_1, \ldots, C_c$ par la règle de décision.

La règle de Bayes est la règle de décision la plus couramment utilisée. Dans le cadre de la théorie bayésienne de la décision, on cherche à minimiser le risque conditionnel :

$$R(C_k|x) = \sum_{l=1}^{c} L(C_k, C_l) P(C_l|x) \tag{1.1}$$

L'élément $L(C_k, C_l)$ de la matrice L désigne le coût associé à une affectation dans la classe C_l alors que la classe réelle est C_k (coût de l'erreur d'affectation).
En classification, on utilise généralement une version binaire de cette règle :

$$L(C_k|C_l) = \begin{cases} 1 & \text{si } k = l \\ 0 & \text{sinon} \end{cases} \tag{1.2}$$

Avec cette fonction de coût, le risque conditionnel devient alors la probabilité conditionnelle de mauvais classement. En minimisant cette probabilité, on obtient la règle du Maximum A Posteriori (MAP) :

$$k^* = arg \min_k R(C_k|x) = arg \max_k P(C_k|x) \tag{1.3}$$

A partir d'un échantillon de taille n, il existe de nombreuses méthodes pour estimer la qualité de l'apprentissage :

- La resubstitution : le jeu de données pour l'apprentissage et le test sont identiques et correspondent à l'ensemble de l'échantillon. Si l'on note n_e, le nombre d'erreurs commises, on aura
$$Erreur = \frac{n_e}{n}$$
- La méthode "Holdout" : une moitié des données est utilisée pour l'apprentissage, l'autre moitié pour le test.
$$Erreur = \frac{n_e}{n/2}$$
- La p-validation croisée : L'échantillon initial est partitionné en p sous-ensembles. On utilise tour à tour chacun de ces sous-ensembles comme jeu de test alors que les (p-1) autres servent pour l'apprentissage. Si l'on note $n_e(m)$ le nombre d'erreurs

de classement commises sur le m-ième sous ensemble, l'erreur est la moyenne des $n_e(m)$

$$Erreur = \frac{\sum_{m=1}^{p} n_e(m)}{n}$$

– La méthode du "Leave-one-out" : la technique du "Leave-one-out" est un cas particulier de la validation croisée pour lequel $p = n$. Cette technique nécessite de relancer n fois la méthode de classification sur $(n-1)$ points ce qui la rend très coûteuse en temps de calcul.

Ces 4 méthodes de validation peuvent être interprétées différemment selon que l'on effectue un apprentissage supervisé ou non supervisé. Concernant la classification automatique, la plupart des auteurs utilisent la méthode de resubstitution car aucune information sur l'étiquette est prise en compte pour l'apprentissage. Par contre, cette méthode est connue pour être optimiste dans le cas supervisé. Les données de test étant identiques aux données utilisées pour l'apprentissage, on ne teste pas la capacité en généralisation mais plutôt la faculté d'apprendre (éventuellement par coeur) sur un jeu particulier. La méthode Holdout est connue pour être pessimiste surtout lorsque la taille de l'ensemble d'apprentissage est faible. Les estimations à partir des deux ensembles de données sont très différentes. A l'inverse, la technique du "Leave-one-out" est sans biais mais réclame un temps de calcul très élevé (souvent prohibitif). Ainsi, on emploie généralement la validation croisée qui fait un compromis entre le biais et la complexité.

1.2 Lien avec d'autres domaines de recherche

Nous avons précédemment défini la reconnaissance des formes comme la recherche de structures dans des données. Cette définition rapproche la RdF de disciplines émergentes comme les réseaux bayésiens ou le data-mining. Dans ces deux cas, il s'agit d'extraire de la connaissance d'un ensemble de données soit par inférence bayésienne, soit par la recherche de règles d'associations. Ces domaines de recherche sont principalement issus de l'Intelligence Artificielle (Informatique) par opposition à la RdF, née de travaux en statistique (Mathématiques). On constate aujourd'hui qu'il existe une fracture entre ces deux disciplines concernant le volume de données à traiter (même si la différence tend à se réduire). La fouille de données, liée à l'Extraction de Connaissances dans des bases de Données (ECD)[2], met en avant les contraintes de gestion et d'efficacité de requêtes dans un très grand volume de données. Il est vrai que la culture informatique et surtout algorithmique pousse à raisonner sur des volumes de données à la limite infinis. Aujourd'hui, certaines bases de données comportent des millions d'entrées pour des milliers de champs. Nous pouvons citer en exemple, l'analyse des fichiers .log de connection à des serveurs WEB. Il est clair que la plupart des méthodes de RdF statistiques soit échouent

[2] Traduction française de KDD ("Knowledge Discovery in Databases")

sur ce type de données, soit sont inapplicables car trop complexes en temps de calcul : les modèles habituellement utilisés pour des dimensions faibles ne sont généralement pas directement transposables dans des dimensions bien supérieures (Ex : Problème de l'inversion des matrices de grande taille). D'un autre côté, on peut s'interroger sur la pertinence de considérer autant de données et de variables. Il est bien évident que sur l'ensemble de variables considérées, seule une infime partie contient de l'information utile. Il peut ainsi être souhaitable d'effectuer un résumé des données par toute méthode adéquate (arbres de décision, rééchantillonnage, ACP[3], etc...) pour diminuer la complexité des données à traiter. On peut alors espérer pouvoir mettre en oeuvre les méthodes statistiques éprouvées de l'analyse de données sur des données de dimension réduite. Il y a fort à parier que le rapprochement actuel des deux disciplines va se poursuivre dans le futur.

1.3 Contexte de l'étude

Le travail présenté dans ce mémoire concerne la classification automatique de données numériques. Le chapitre 2 en présente la plupart des approches, sans toutefois prétendre être exhaustif. Nous avons choisi de distinguer les méthodes paramétriques des méthodes non paramétriques car c'est dans le cadre des premières que se situent les apports essentiels de cette thèse.

Dans la problématique de la classification automatique, nous cherchons à répondre aux deux questions suivantes : "Peut-on pallier l'échec de la plupart des méthodes de classification automatique en présence de données aberrantes ?" et "Peut-on améliorer la qualité de l'apprentissage en ajoutant un minimum de données étiquetées ?". Ces thématiques constituent deux préoccupations actuelles majeures de l'analyse de données et de tout processus d'apprentissage.

Les données aberrantes peuvent, aujourd'hui, avoir des causes aussi diverses que multiples (mutation d'un gène, donnée étrangère au problème considéré, défaut d'un capteur, etc...). Certains autres problèmes spécifiques au traitement informatique des données (en particulier audiovisuelles) sont rappelés dans la dernière partie du chapitre 2. Il est donc indispensable de comprendre les raisons de l'échec de la plupart des algorithmes de classification automatique en présence de ce type de données et d'y apporter des solutions. Dans la première partie du chapitre 3, nous donnons la raison de cet échec et présentons les principales approches de la classification automatique robuste que sont : l'usage des statistiques robustes et en particulier les M-estimateurs permettant de réduire l'influence des points aberrants, l'addition d'une classe particulière dédiée au traitement des points aber-

[3]Pour l'ACP, il existe d'ores et déjà une technique basée sur l'algorithme EM qui évite de calculer l'ensemble des valeurs propres la matrice de covariance, ce qui constitue un gain substantiel[Row98]

rants, la définition d'un modèle explicite des données contaminées par les points aberrants.

La seconde thématique de notre étude concerne la supervision partielle. Les données non étiquetées sont généralement disponibles en très grand nombre car faciles à obtenir et peu coûteuses (Ex. : acquisition en continu d'images). A l'inverse, les données supervisées sont plus rares car elles nécessitent une intervention humaine ou logistique onéreuse voire dangereuse (Ex. : prospection pétrolière). L'idée de la supervision partielle est ainsi de profiter au mieux d'une expertise sur un ensemble même réduit de données pour améliorer un apprentissage. De nouvelles approches de l'apprentissage, comme les méthodes coopératives[4] ou de fusion, concernent des problématiques proches de la supervision. Dans le cadre de cette étude, nous dressons un état de l'art de la semi-supervision pour la classification. Là encore, divers algorithmes sont présentés, qu'ils soient fondés sur une approche paramétrique ou non.

Une nouvelle méthode de classification paramétrique robuste partiellement supervisée est proposée au chapitre 4. Elle repose sur l'estimation itérative des paramètres d'un mélange, inspirée de l'algorithme EM. Chaque composante de ce mélange suit un modèle de contamination. L'originalité de la méthode réside dans une combinaison d'estimation robuste et non robuste de ce modèle. Celui-ci permet d'introduire de manière assez évidente une option de rejet (en distance) des points aberrants selon une règle d'affection originale elle aussi. Une extension naturelle à la supervision partielle est proposée.

A la fin de ce mémoire, nous concluons ces travaux de recherche, donnons quelques pistes pour les améliorations toujours possibles et dressons quelques perspectives scientifiques.

[4] On peut citer en exemple le co-training où plusieurs discriminateurs tentent d'apprendre le même concept avec des données de nature différente [BM98b]. Par exemple, pour un problème de reconnaissance du locuteur à la télévision, on peut travailler séparément sur les images (reconnaissance visuelle) et le son (reconnaissance vocale). Si l'un des modules identifie clairement le locuteur, il peut aider (superviser) l'autre.

Chapitre 2

Des approches de la classification automatique

Après une présentation générale de ce qu'est la reconnaissance de formes, ce chapitre a pour objectif de décrire la problématique de la classification automatique. Les deux premières parties décrivent plusieurs méthodes de classification en faisant la distinction entre les méthodes paramétriques et les méthodes non paramétriques. Un accent particulier a été mis sur l'approche paramétrique et les algorithmes de type EM qui constituent une des bases de notre travail. La troisième partie relate les principaux problèmes rencontrés en classification automatique et les solutions éventuelles pour y remédier. Le problème particulier des données aberrantes sera discuté dans le chapitre suivant.

2.1 Rappel sur la complexité de la classification

Dans cette partie, nous rappelons quelques résultats d'informatique théorique et de complexité concernant le problème de la classification. Celui-ci peut se définir sous une forme simplifiée comme :
Classification : Étant donné un ensemble de points $\chi = \{x_1, \ldots, x_n\}$, une distance $d(x, y) \in Z_0^+$ définie pour tout couple de points (x_i, x_j), deux entiers positifs b et c.
Problème : *Existe t'il une partition* $C_1, \ldots, C_c$ *de* χ *en* c *classes telle que* $\forall k \in [1, c]$ *et* $\forall (x, y) \in C_k$, $d(x, y) < b$ *?*

On peut essayer de résoudre ce problème par une méthode brutale en engendrant l'ensemble des partitions possibles puis en testant la validité de la partition. Cependant, la taille de l'espace des partitions possibles est de l'ordre de $\mathcal{O}(\frac{n^k}{k!})$. L'algorithme proposé est donc exponentiel. Mais, chacune des partitions doit-elle être examinée ? Prenons, par exemple, une instance de ce problème de classification pour 5 points en dimension 1 (voir Fig. (2.1)). Dans cette figure, la partition de gauche contient un chevauchement des classes. On peut se demander si cette partition doit être testée sachant que celle de droite l'est.

L'exigence d'une structure, un voisinage, entre les points doit pouvoir guider la recherche d'une solution au problème.

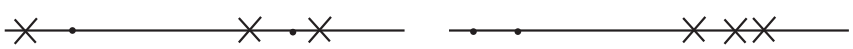


FIG. 2.1 – La partition de gauche mérite t'elle d'être examinée sachant que la partition de droite l'est ?

Cette notion de voisinage entre les points dépend la dimension de l'espace dans lequel ils sont. En dimension 1, par exemple, on voit bien que la recherche des plus proches voisins peut-être réduite à un choix parmi les deux voisins (gauche et droite). La dimension joue ainsi un rôle essentiel dans la complexité de ce problème. Il est clair que la classification est un problème appartenant à la classe de complexité NP [GJ79]. En effet, il est facile de vérifier polynomialement qu'une partition en c classes répond ou non au problème posé (certificat polynomial). Il suffit pour cela de vérifier que pour chacune des classes C_k tout couple de points respecte la contrainte sur la distance. Mais comme souvent, il existe un saut de classe de complexité lorsque l'on passe d'un problème de 2 à plus de 3 dimensions (de polynomial à NP-complet). Pour démontrer la NP-complétude, il est nécessaire de transformer une instance du problème de la classification en une instance d'un problème NP-complet connu [Hab98]. Cette transformation a été établie pour divers problèmes standards NP-complets de la théorie des graphes tels que la 3-coloration, la coupe maximale ou la couverture maximale [Bru78]. Le problème de la recherche de la partition, et non plus l'existence de celle-ci, est un problème NP-difficile. Ainsi, tout algorithme de classification permettant d'utiliser des données en dimensions quelconques est une heuristique d'un problème NP-difficile.

2.2 L'approche paramétrique en classification

Les données observées sont supposées être des réalisations d'un vecteur aléatoire X de loi inconnue P. L'objectif est de reconstituer P à partir de ses réalisations. Pour cela, on se donne un ensemble de lois possibles, $\{P_\theta, \theta \in \Theta\}$ en supposant qu'il existe $\theta^* \in \Theta$ tel que $P = P_{\theta^*}$. Il s'agit de déterminer $P_{\hat{\Theta}(X)}$ tel que $P = P_{\hat{\Theta}(X)}$.

On parlera d'approche paramétrique lorsque Θ est une partie d'un espace euclidien[5] (Ex. : $\Theta = \{N(\mu, \sigma) \backslash \mu, \sigma \in \Re\}$). L'approche est qualifiée de non paramétrique si Θ est un espace fonctionnel (Ex. : fonctions noyaux). Ces définitions soulèvent immédiatement deux problèmes :

[5] Un espace euclidien est un espace vectoriel de dimension finie muni d'un produit scalaire.

– comme P est inconnue, rien ne garantit qu'il appartienne à la famille $\{P_\theta, \theta \in \Theta\}$. Par exemple, on recherche une densité P_θ unimodale alors que P est multimodale ;
– Le modèle supposé contient trop de paramètres, on dispose d'un nombre insuffisant d'observations (voir problème de la dimensionnalité des données, section 2.4.3).

La décomposition par un modèle de mélange est une approche paramétrique communément adoptée dans le contexte de la classification.

2.2.1 Modèle de mélange

Comme nous l'avons vu précédemment, la densité de probabilité $f(x)$ en un point x est inconnue. Le principe d'un modèle de mélange est de décomposer cette densité en une somme de c composantes $f(\bullet|\Theta_k)$ correspondant aux c classes dont on va estimer les paramètres Θ_k $(k = 1, \ldots, c)$ à partir d'un échantillon χ.

χ est un ensemble de n réalisations x_i d'un vecteur aléatoire X de dimension p et s'écrit :

$$\chi = \{x_1, \ldots, x_n\}$$

Ces densités de probabilités $f(\bullet|\Theta_k)$ peuvent aller du modèle le plus simple aux distributions multi-modales les plus complexes. Les proportions π_k entre les différentes composantes représentent les probabilités a priori des différentes classes. En général, ces proportions sont également inconnues et l'on doit les estimer sous les contraintes :

$$\pi_k \in]0; 1[\text{ et } \sum_{k=1}^{c} \pi_k = 1$$

Ainsi, on notera $\Theta = [\pi_1, \ldots, \pi_c, \Theta_1, \ldots, \Theta_c]^T$ le vecteur de paramètres à estimer. La densité de probabilité $f(x)$ en un point x est ainsi approximée conditionnellement aux paramètres Θ par $f(x|\Theta)$:

$$f(x|\Theta) = \sum_{k=1}^{c} \pi_k f(x|\Theta_k) \tag{2.1}$$

En classification supervisée, on dispose d'une information supplémentaire concernant l'appartenance des points aux classes. Celle-ci permet d'estimer directement les paramètres Θ du modèle. A savoir :

– les proportions des classes, s'il n'y a pas de biais de sélection ;
– les paramètres Θ_k des classes avec le problème du choix du bon modèle.

Par la suite, l'estimation des paramètres d'un mélange sera décrite dans le contexte de l'algorithme EM. Cependant, cette estimation n'a de sens que si le modèle Θ est identifiable [MP00]. Une famille de densité $f(x|\Theta)$ est identifiable si pour deux valeurs distinctes θ_1 et θ_2 du paramètre Θ, les densités $f(x|\theta_1)$ et $f(x|\theta_2)$ sont distinctes :

$$\theta_1 = \theta_2 \Longleftrightarrow f(x|\theta_1) = f(x|\theta_2) \tag{2.2}$$

Cette définition doit être modifiée dans le cadre d'un modèle de mélange de paramètre Θ. Supposons que le mélange $f(x|\Theta)$ soit composé de deux densités $f(x|\Theta_k)$ et $f(x|\Theta_l)$ appartenant à la même famille de densité. Si l'on pose $\Theta_1 = [\pi_k, \pi_l, \Theta_k, \Theta_l]^T$ et $\Theta_2 = [\pi_l, \pi_k, \Theta_l, \Theta_k]^T$, on obtient $f(x|\Theta_1) = f(x|\Theta_2)$ mais $\Theta_1 \neq \Theta_2$ du fait de l'inversion des composantes. La définition précédente de l'identifiabilité est étendue pour permettre une permutation entre les composantes. Ce problème plus connu sous le nom de "label-switching" intervient essentiellement dans l'approche bayésienne pour les modèles de mélange [Ste00].

La plupart des mélanges finis de densités continues sont identifiables à l'exception notable des densités uniformes. En effet, une densité uniforme $f(x|U)$ sur un intervalle I possède une infinité de décomposition par un mélange fini de densités uniformes sur le même intervalle :

$$f(x|U) = \sum_{k=1}^{K} \pi_k f(x|U_k)$$

Les mélanges de composantes normales sont identifiables.

2.2.2 Cas des mélanges gaussiens

Les mélanges gaussiens sont couramment utilisés en classification car ils correspondent souvent à la loi de distribution d'une variable (Ex. : taille d'une personne). De plus, ils sont relativement bien maîtrisés et l'on dispose de procédures efficaces pour les manipuler.

Dans le cas gaussien, les Θ_k représentent les moyennes et les matrices de covariance de la k-ième classe que l'on note μ_k et Σ_k. La densité de probabilité multi-dimensionnelle en un point x conditionnellement à Θ_k s'écrit donc :

$$f(x|\Theta_k) = \frac{1}{(2\pi)^{p/2}\sqrt{|\Sigma_k|}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \tag{2.3}$$

où $|\Sigma_k|$ désigne le déterminant de la matrice de covariance de dimension $p \times p$.

La figure (2.2) illustre un mélange de 3 gaussiennes en proportions égales (à gauche) et l'histogramme correspondant (à droite). La courbe en pointillé à gauche représente la densité de probabilité théorique du mélange résultant en tenant compte des proportions.

2.2.2.1 Décomposition de la matrice de covariance

La matrice de covariance Σ est une matrice symétrique définie positive de dimension $p \times p$. Dans sa version la plus générale (non contrainte), l'estimation de cette matrice nécessite d'estimer $\frac{(p+1)p}{2}$ coefficients (partie triangulaire supérieure ou inférieure).

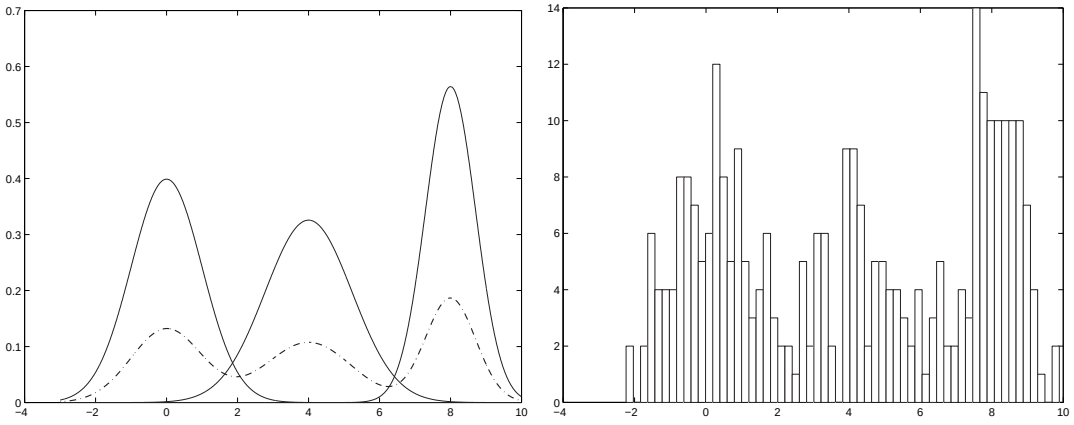


FIG. 2.2 – (A gauche) un mélange de 3 gaussiennes (en pointillé) et leurs densités théoriques (trait plein) - (A droite) l'histogramme associé

Dans [CG95], Celeux et Govaert ont proposé de décomposer la matrice de covariance selon ses valeurs spectrales en considérant 14 modèles contraints. Cette décomposition peut s'écrire sous la forme :

$$\Sigma_k = \lambda_k D_k A_k D_k^T \tag{2.4}$$

Les différents cas sont obtenus par des contraintes simultanées ou non des λ_k, D_k, A_k qui permettent de faire varier le volume, sa forme (sphérique ou elliptique) et l'orientation de la matrice de covariance. Ces modèles contraints sont également qualifiés de modèles parcimonieux. La table suivante récapitule les hypothèses les plus utilisées concernant la matrice de covariance :

Σ_k	Type	Volume	Forme	Orientation
λI	Sphérique	constant	constant	non définie
$\lambda_k I$	Sphérique	variable	constant	non définie
$\lambda D A D^T$	Elliptique	constant	constant	constante
$\lambda D_k A D_k^T$	Elliptique	constant	constant	variable
$\lambda_k D_k A D_k^T$	Elliptique	variable	constant	variable
$\lambda_k D_k A_k D_k^T$	Elliptique	variable	variable	variable

Les différentes contraintes sur la matrice de covariance diminuent la complexité du modèle à apprendre mais également le pouvoir d'expression de celui-ci. Dans ce travail, nous nous sommes principalement intéressés aux modèles non contraints.

L'estimation du vecteur de paramètres Θ d'un mélange gaussien peut s'effectuer à l'aide de l'algorithme EM. Le principe de cet algorithme va être présenté maintenant. Son application pour les modèles de mélange est décrite par la suite.

2.2.3 L'algorithme Expectation-Maximization (EM)

2.2.3.1 Le cadre général

L'algorithme Expectation-Maximization (EM) est une technique itérative de maximisation de la loi de vraisemblance en présence de données incomplètes. On l'attribut généralement à Dempster, Laird et Rubin [DLR77] même si ces mêmes auteurs l'imputent à Hartley [Har58].

La loi de vraisemblance des données relativement au modèle de paramètre Θ s'écrit :

$$\mathcal{L}(\Theta) = P(X|\Theta) \tag{2.5}$$

Sous l'hypothèse que les données de l'ensemble d'apprentissage $\chi = \{x_1, \ldots, x_n\}$ sont des réalisations indépendantes du vecteur aléatoire X, la loi de vraisemblance se réécrit en un produit de probabilités :

$$\mathcal{L}(\Theta) = \prod_{i=1}^{n} f(x_i|\Theta) \tag{2.6}$$

Dans le cadre des modèles de mélange, cette équation se développe sous la forme :

$$\mathcal{L}(\Theta) = \prod_{i=1}^{n} \sum_{k=1}^{c} \pi_k f(x_i|\Theta_k) \tag{2.7}$$

dont le logarithme s'écrit :

$$\log \mathcal{L}(\Theta) = \sum_{i=1}^{n} \log \sum_{k=1}^{c} \pi_k f(x_i|\Theta_k) \tag{2.8}$$

Selon l'estimation par la méthode du maximum de vraisemblance, cela équivaut à rechercher les racines de l'équation :

$$\frac{\partial log \mathcal{L}(\Theta)}{\Theta} = 0 \tag{2.9}$$

Cependant, le vecteur aléatoire X n'est qu'une observation partielle du phénomène considéré. La maximisation de l'équation (2.8) étant difficile à réaliser directement, on introduit une variable aléatoire Z correspondant aux données manquantes cachées. La connaissance de ces données cachées permettrait d'utiliser de manière simple le principe du maximum de vraisemblance. L'idée de l'algorithme EM est ainsi de faciliter le processus d'optimisation en utilisant une estimation de ces données manquantes.

On appellera ainsi $Y = (X, Z)$ les données complètes. Au lieu de maximiser $\mathcal{L}(\Theta)$, on maximise itérativement l'espérance conditionnelle de la vraisemblance complète qui s'écrit :

$$\mathcal{L}_c(\Theta|Y) = P(X, Z|\Theta)$$

L'algorithme EM alterne successivement deux phases décrites dans l'algorithme 1. L'une des proprietés de cet algorithme est d'améliorer la vraisemblance $\mathcal{L}(\Theta)$ des paramètres après chaque itération ce qui permet de démontrer sa convergence (voir démonstration p. 40 [Dan98]).

Algorithme 1: L'algorithme EM standard

Entrée : $\theta^{(0)}$ une valeur initiale des paramètres du modèle , χ un ensemble d'observations, ϵ seuil pour la convergence de l'algorithme

Sortie : Un maximum (local) Θ^* de la loi de vraisemblance

$t \leftarrow 0$;

Initialisation du modèle $\hat{\Theta}^{(0)} = \theta^{(0)}$;

répéter

(E-Step) Calcul de l'espérance conditionnelle de la vraisemblance complète :

$$Q(\Theta|\hat{\Theta}^{(t)}) = E[\mathcal{L}_c(\Theta)|X, \hat{\Theta}^{(t)}]$$

(M-Step) Maximisation de $Q(\Theta|\hat{\Theta}^{(t)})$:

$$\hat{\Theta}^{(t+1)} = \arg\max_{\Theta} Q(\Theta|\hat{\Theta}^{(t)})$$

$t \leftarrow t + 1$;

jusqu'à $||Q(\Theta|\hat{\Theta}^{(t)}) - Q(\Theta|\hat{\Theta}^{(t-1)})|| < \epsilon$;

2.2.3.2 EM pour les modèles de mélange

L'algorithme EM est abondamment utilisé en classification dans le cadre d'un modèle de mélange [MP98][FR98a]. Comme nous l'avons dit plus haut, la log-vraisemblance $\log \mathcal{L}(\Theta)$ s'écrit :

$$\log \mathcal{L}(\Theta) = \sum_{i=1}^{n} \log \sum_{k=1}^{c} \pi_k f(x_i; \Theta_k)$$

Cette maximisation serait plus simple si l'on connaissait la classe d'appartenance de tous les points. On remarquera au passage que cela correspondrait à effectuer une classification en mode totalement supervisé. On introduit un vecteur aléatoire supplémentaire Z correspondant au vecteur d'appartenance d'un point x_i à la classe C_k. Généralement, les réalisations $z_i = (z_{i,1}, \ldots, z_{i,c})$ de Z sont prises comme des réalisations indépendantes et identiquement distribuées d'une distribution multinomiale $Mult_k(1, \pi)$ où $\pi = (\pi_1, \ldots, \pi_c)^T$.

On écrit pour l'élément x_i :

$$z_{ik} = \begin{cases} 1 & \text{si } x_i \text{ appartient à la classe } C_k \\ 0 & \text{sinon} \end{cases}$$

Sous ces conditions, la log-vraisemblance complète s'écrit :

$$\begin{aligned} \log \mathcal{L}_c(\Theta|Y) &= \log(P(X, Z|\Theta)) \\ &= \log(\prod_{i=1}^{n} f(x_i, z_i|\Theta)) \\ &= \log(\prod_{i=1}^{n}\prod_{k=1}^{c}(\pi_k f(x_i|\Theta_k))^{z_{ik}}) \\ \log \mathcal{L}_c(\Theta|Y) &= \sum_{i=1}^{n}\sum_{k=1}^{c} z_{ik} \log(\pi_k f(x_i; \Theta_k)) \end{aligned} \tag{2.10}$$

Algorithme 2: L'algorithme EM pour les modèles de mélange

Entrée : c le nombre de composantes du mélange, $\theta^{(0)} = [\pi_1^{(0)}, \ldots, \pi_c^{(0)}, \theta_1^{(0)}, \ldots, \theta_1^{(0)}]$ une valeur initiale des paramètres du modèle, χ un ensemble d'observations, ϵ le seuil pour la convergence de l'algorithme

Sortie : Un maximum (local) Θ^* de la loi de vraisemblance

$t \leftarrow 0$;

Initialisation du modèle $\hat{\Theta}^{(0)} = \theta^{(0)}$;

répéter

(E-Step) Calcul des probabilités a posteriori $\hat{z}_{ik}^{(t)}$:

$$\hat{z}_{ik}^{(t)} = \frac{\hat{\pi}_k^{(t)} f(x_i; \hat{\Theta}_k^{(t)})}{\sum_{l=1}^{c} \hat{\pi}_l^{(t)} f(x_i; \hat{\Theta}_l^{(t)})}$$

(M-Step) Maximisation de $Q(\Theta|\hat{\Theta}^{(t)})$:

$$\hat{\Theta}^{(t+1)} = \arg\max_{\Theta} Q(\Theta|\hat{\Theta}^{(t)})$$

$t \leftarrow t + 1$;

jusqu'à $||Q(\Theta|\hat{\Theta}^{(t)}) - Q(\Theta|\hat{\Theta}^{(t-1)})|| < \epsilon$;

D'après l'équation (2.10), la log-vraisemblance complète est une fonction linéaire des z_{ik}. L'étape d'estimation de cette vraisemblance se résume donc à remplacer les z_{ik} par leur espérance. Soit $\hat{Z}_{ik}$ l'estimateur de cette espérance et $\hat{z}_{ik}$ une réalisation de celui-ci.

On a

$$\hat{Z}_{ik}^{(t)} = E[Z_{ik}|X = x_i, \hat{\Theta}^{(t)}] \tag{2.11}$$

$$\hat{z}_{ik}^{(t)} = P(Z_{ik} = 1|X = x_i, \hat{\Theta}^{(t)}) = \frac{\hat{\pi}_k^{(t)} f(x_i; \hat{\Theta}_k^{(t)})}{\sum_{l=1}^{c} \hat{\pi}_l^{(t)} f(x_i; \hat{\Theta}_l^{(t)})} \tag{2.12}$$

L'algorithme 2 décrit le fonctionnement de l'algorithme EM pour les modèles de mélange.

2.2.3.3 EM pour un mélange gaussien

Dans l'algorithme précédent, la phase de maximisation ne peut être décrite car elle dépend du modèle théorique choisi pour les classes. L'algorithme 3 exprime celle-ci dans le cas d'un mélange de composantes gaussiennes non contraintes.

En pratique, cet algorithme fournit de bons résultats même s'il n'échappe pas aux principaux problèmes des algorithmes de classification :

- L'algorithme EM converge de manière sûre vers un optimum éventuellement local. Le résultat final dépend de l'initialisation.
- On doit fixer le nombre de classes cherchées via le nombre de composantes estimées.
- La convergence de l'algorithme peut être très lente.
- L'algorithme peut échouer lorsque le système est mal conditionné (Exemple : inversion de matrices singulières).
- Il est sensible aux données aberrantes (voir chapitre 3).

Nous reparlerons de la plupart de ces problèmes par la suite et des solutions que l'on peut éventuellement y apporter (voir section 2.4).

La complexité en temps de calcul de cet algorithme dépend essentiellement du modèle de classe choisi. On peut remarquer que l'estimation de la probabilité a priori d'un point pour une classe gaussienne nécessite de calculer la distance de Mahalanobis ; ceci requérant une multiplication vecteur-matrice, la complexité de cette opération est quadratique en p. Si l'on utilise une matrice intermédiaire de stockage des probabilités a priori, l'étape E-step de l'algorithme EM est de complexité $\Theta(ncp^2)$ pour une complexité en espace de $\Theta(nc)$. Le calcul des probabilités a priori des classes nécessite un simple parcours des probabilités a posteriori des points. On en déduit que ce calcul nécessite $\Theta(nc)$ opérations pour l'ensemble des classes. La mise à jour des vecteurs moyenne est de la même nature, si ce n'est que l'on doit considérer la dimension des vecteurs ; il en résulte une complexité de $\Theta(ncp)$. La mise à jour des covariances est effectuée en $\Theta(ncp^2)$. Enfin, on doit inverser les matrices de covariance. Ces matrices étant symétriques définies positives, il est préférable d'utiliser la décomposition de Choleski à la place de la décomposition LU (Lower-Upper). Cela permet de limiter quelque peu les erreurs numériques d'arrondi. La complexité de telles décompositions est en $\mathcal{O}(p^3)$ où p est la taille du vecteur-forme. Comme l'on doit

Algorithme 3: L'algorithme EM dans le cas des mélanges gaussiens

Entrée : c le nombre de composantes du mélange, χ un ensemble d'observations, $\theta^{(0)} = [\pi_1^{(0)}, \ldots, \pi_c^{(0)}, \mu_1^{(0)}, \Sigma_1^{(0)}, \ldots, \mu_c^{(0)}, \Sigma_c^{(0)}]$ une valeur initiale des paramètres du modèle, ϵ le seuil pour la convergence de l'algorithme

Sortie : Un maximum (local) Θ^* de la loi de vraisemblance

$t \leftarrow 0$;

Initialisation du modèle $\hat{\Theta}^{(0)} = \theta^{(0)}$;

répéter

(E-Step) Calcul des probabilités a posteriori $\hat{z}_{ik}^{(t)}$:

$$\hat{z}_{ik}^{(t)} = \frac{\hat{\pi}_k^{(t)} f(x_i; \hat{\Theta}_k^{(t)})}{\sum_{l=1}^{c} \hat{\pi}_l^{(t)} f(x_i; \hat{\Theta}_l^{(t)})}$$

(M-Step) Estimation des π_k, μ_k, Σ_k maximisant $Q(\Theta|\hat{\Theta}^{(t)})$

Estimation de la probabilité a priori $\hat{\pi}_k$ de la k-ième classe :

$$\hat{\pi}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}}{n} \tag{2.13}$$

Estimation de la moyenne $\hat{\mu}_k$ de la k-ième classe

$$\hat{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} x_i}{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}} \tag{2.14}$$

Estimation de la matrice de covariance $\hat{\Sigma}_k$ de la k-ième classe

$$\hat{\Sigma}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} (x_i - \hat{\mu}_k^{(t+1)})(x_i - \hat{\mu}_k^{(t+1)})^T}{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}} \tag{2.15}$$

$t \leftarrow t + 1$;

jusqu'à $||Q(\Theta|\hat{\Theta}^{(t)}) - Q(\Theta|\hat{\Theta}^{(t-1)})|| < \epsilon$;

inverser chacune des matrices, cette opération est en $\mathcal{O}(cp^3)$. On peut légitiment supposer que le nombre de points n est très grand par rapport à la taille p du vecteur-forme [6]. Ainsi, la complexité de l'algorithme EM dans le cas gaussien (matrices de covariance non contraintes) est $\Theta(tncp^2)$ où t est le nombre d'itérations effectuées.

2.2.4 Quelques variantes d'EM

L'algorithme EM possède de nombreuses variantes cherchant chacune à pallier les divers problèmes évoqués précédemment. Dans cette partie, nous allons en présenter trois parmi celles nous semblant les plus pertinentes dans le contexte de cette étude : les algorithmes SEM, CEM et "Component-Wise EM".

2.2.4.1 Stochastic EM

L'algorithme SEM est une variante stochastique de l'algorithme EM [CD85]. Une étape de simulation est ajoutée entre les phases d'estimation et de maximisation dans le but de le rendre moins sensible aux minima locaux. Celle-ci perturbe le système en effectuant une classification aléatoire suivant les probabilités a posteriori estimées précédemment. On engendre ainsi une partition aléatoire stricte par un tirage aléatoire biaisé selon la loi multinomiale de paramètres $Mult_k(1; \hat{z}_{i,1}, \ldots, \hat{z}_{i,c})$. L'algorithme s'affranchit également du problème du nombre de classes en faisant décroître celui-ci à partir d'une valeur majorée c_{max} : les classes ayant peu de points sont éliminées avant un redémarrage. Les détails de SEM sont décrits par l'algorithme 4.

Après une phase initiale d'échauffement où le nombre de classes décroît rapidement, on observe un régime stationnaire de la suite des paramètres estimés $\{\Theta^{(t)}\}$ [CD85]. En calculant la variance des éléments de cette suite, on peut évaluer la fiabilité de l'estimation produite. L'estimation finale de Θ peut être effectuée en moyennant les éléments $\{\Theta^{(t)}\}$ du régime stationnaire. Les auteurs ont montré empiriquement que les estimations obtenues par SEM sont de moins bonne qualité que celles issues de l'algorithme EM lorsque les données sont en faibles quantités. Dans ce cas, l'étape de simulation peut amener à prendre en compte dans l'estimation des points très éloignés avec une influence non négligeable. Cet algorithme peut être utilisé comme point de départ de l'algorithme EM [CCD95]. La complexité de SEM supérieure à celle de l'algorithme EM, d'une part parce que l'on rajoute une étape de simulation (S-Step), d'autre part parce que l'on peut choisir un nombre supérieur de classes que l'on fait décroître progressivement (accélération).

[6] Dans le cas contraire, on dispose d'un nombre insuffisant de points pour estimer l'ensemble des coefficients de la matrice de covariance. Il est nécessaire de disposer d'au moins $\frac{(p+1)*p}{2}$ points.

Algorithme 4: L'algorithme SEM

Entrée : $\theta^{(0)}$ une valeur initiale des paramètres du modèle, un ensemble d'observations χ, c_{max} le nombre maximal de classes, n_{min} le nombre de points minimal par classe, t_{max} le nombre maximal d'itérations

Sortie : Une suite stationnaire $\{\Theta^{(t)}\}$ d'estimation des paramètres du modèle

$t \leftarrow 0$;

Initialisation du modèle $\hat{\Theta}^{(0)} = \theta^{(0)}$;

$c \leftarrow c_{max}$;

répéter

(E-Step) Calcul des probabilités a posteriori $\hat{z}_{ik}^{(t)}$:

$$\hat{z}_{ik}^{(t)} = \frac{\hat{\pi}_k^{(t)} f(x_i; \hat{\Theta}_k^{(t)})}{\sum_{l=1}^{c} \hat{\pi}_l^{(t)} f(x_i; \hat{\Theta}_l^{(t)})}$$

(S-Step) Simulation de la partition à partir de la distribution a posteriori $P(Z|\chi, \Theta^{(t)})$

$$u_{ik}^{(t)} = \begin{cases} 1 & \text{si la classe k est tirée au sort selon la loi multinomiale } (\hat{z}_{i,1}^{(t)}, \ldots, \hat{z}_{i,c}^{(t)}) \\ 0 & \text{sinon} \end{cases}$$

// Suppression des classes ayant un effectif trop faible

si $\exists\ k\ tel\ que\ \sum_{i=1}^{n} u_{ik} \leq n_{min}$ **alors**

Redémarrer avec c-1 classes ;

(M-Step) Maximisation de la vraisemblance :

$$\Theta^{(t+1)} = \arg\max_{\Theta}(\sum_{i=1}^{n}\sum_{k=1}^{c} u_{ik}^{(t)} \log(\hat{\pi}_k^{(t)} f(x_i; \hat{\Theta}_k^{(t)})))$$

$t \leftarrow t + 1$;

jusqu'à $t < t_{max}$;

2.2.4.2 Classification EM

L'algorithme CEM (Classification EM) est une autre variante de l'algorithme EM spécifique à la classification [CG92]. On recherche simultanément les paramètres Θ du modèle et la partition U. Le critère à maximiser devient alors la log-vraisemblance classifiante [Sym81] :

$$CL(\Theta, U) = \sum_{i=1}^{n}\sum_{k=1}^{c} u_{ik} \log(\pi_k f(x_i; \Theta_k)) \tag{2.16}$$

où U désigne la matrice de partition stricte d'éléments u_{ik}. Ce critère s'exprime comme la log-vraisemblance complète vue précédemment (voir Eq. (2.10)). Dans l'étape de maximisation, CEM met à jour les paramètres de chaque classe à partir du sous-ensemble des

Algorithme 5: L'algorithme CEM pour les modèles de mélange

Données : $\theta^{(0)}$ une valeur initiale des paramètres du modèle, χ un ensemble d'observations

Sortie : Θ^* et U^* maximisant la vraisemblance classifiante

$t \leftarrow 0$

Initialisation du modèle $\hat{\Theta}^{(0)} = \theta^{(0)}$;

répéter

E-Step (Estimation)

Calcul des probabilités a posteriori $\hat{z}_{ik}^{(t)}$:

$$\hat{z}_{ik}^{(t)} = \frac{\pi_k^{(t)} f(x_i; \Theta_k^{(t)})}{\sum_{l=1}^{c} \pi_l^{(t)} f(x_i; \Theta_l^{(t)})} \tag{2.17}$$

C-Step (Classification)

Affectation selon le critère MAP :

$$u_{ik}^{(t)} = \begin{cases} 1 & \text{si } \hat{z}_{ik}^{(t)} = \max_l(\hat{z}_{il}^{(t)}) \\ 0 & \text{sinon} \end{cases}$$

M-Step (Maximisation)

Maximisation de la vraisemblance classifiante :

$$\Theta^{(t+1)} = \arg\max_{\Theta} CL(\Theta^{(t)}, U^{(t)})$$

$t \leftarrow t + 1$

jusqu'à *Stabilité de la partition* U;

points qui lui ont été affectés. On peut remarquer que si l'on choisit le modèle gaussien avec $\Sigma_k = I$ et $\pi_k = \frac{1}{c}$, cet algorithme est similaire à celui des centres mobiles [DH73]. Il donne de bons résultats dans le cas de classes bien séparées, mais produit des estimations biaisées lorsqu'il y a du mélange [CG92]. En effet, les centres des classes sont repoussés par les zones de mélange à cause de l'affectation stricte. D'un point de vue pratique, cet algorithme converge beaucoup plus rapidement que la version standard.

2.2.4.3 Component-Wise EM

L'algorithme Component-Wise EM a été développé dans le but d'accélérer la convergence de l'algorithme EM [CCFM99]. L'une des causes possibles de cette lenteur réside dans la réalisation simultanée de contraintes antagonistes à chaque itération. L'idée de cet algorithme est de mettre à jour les paramètres d'une seule composante à la fois, laissant les autres inchangées. Bien évidemment, toutes les composantes sont cycliquement mises

à jour. L'une des principales difficultés pour cet algorithme est de parvenir à respecter la

Algorithme 6: L'algorithme "Component-Wise EM"

Données : Une valeur initiale des paramètres du modèle $\Theta^{(0)}$, un ensemble d'observations χ

Sortie :

$t \leftarrow 0$

Initialisation du modèle $\Theta^{(0)}$

répéter

// Sélection de la composante à maximiser

$k \leftarrow t - \lfloor \frac{t}{c} \rfloor * c + 1$

E-Step (Estimation)

// Calcul des probabilités a posteriori pour la k-ème composante :

pour *i = 1 à n* **faire**

$$\hat{z}_{ik}^{(t)} = \frac{\pi_k^{(t)} f(x_i; \Theta_k^{(t)})}{\sum_{l=1}^{c} \pi_l^{(t)} f(x_i; \Theta_l^{(t)})}$$

M-Step (Maximisation)

// Maximisation de la vraisemblance classifiante :

$$\Theta_k^{(t+1)} = \arg\max_{\Theta} CL(\Theta^{(t)}, U^{(t)})$$

pour *l = 1 à c* **faire**

$\Theta_l^{(t+1)} = \Theta_l^{(t)}$ si $l \neq k$

$t \leftarrow t + 1$

jusqu'à *Condition arret*;

contrainte : $\sum_{l=1}^{c} \pi_l^{(t)} = 1$. Néanmoins, les auteurs ont démontré que celui-ci possède les mêmes propriétés de convergence que l'algorithme EM. Le gain en rapidité n'est obtenu que lorsque l'algorithme EM converge lentement, c'est à dire pour des données mélangées.

2.2.4.4 Encore des variantes...

Parmi les autres variantes de l'algorithme EM, citons :

- SMEM (Split Merge EM) [UNGH99] ; Pour éviter des minima locaux, on peut effectuer des divisions-fusions successives des classes fondées sur des distances entre elles (Ex. : Divergence de Kullback-Liebler).
- SAEM (Simulated Annealing EM) [CCD95] ; Cet algorithme effectue un compromis entre les algorithmes EM et SEM. L'estimation des paramètres est un mélange des

paramètres estimés par EM et SEM :

$$\Theta^{(t+1)} = (1 - \gamma^{(t+1)})\Theta_{EM}^{(t+1)} + \gamma^{(t+1)}\Theta_{SEM}^{(t+1)}$$

où γ est un terme décroissant allant de 1 vers 0 assimilable à la température d'un recuit simulé.

- MCEM (Monté-Carlo EM) [WT90] ; Lorsque l'espérance de la vraisemblance complète est trop complexe pour être exprimée directement, on peut approcher celle-ci par simulation numérique. Ainsi, on engendre m réalisations des données manquantes selon la distribution $p(Z|X, \Theta)$ dont on fait la moyenne. L'étape de maximisation est effectuée sur cette espérance simulée. Lorsque m est grand, MCEM se comporte comme EM si ce n'est que la propriété sur la croissance monotone de la vraisemblance est perdue.
- NEM (Neighborhood EM) [Amb96] ; L'algorithme NEM est une variante de EM pour les données spatiales (données avec des contraintes de voisinage). La vraisemblance est augmentée d'un terme relatif au voisinage entre les points. Cet algorithme peut être utilisé pour la segmentation d'images (classification de pixels).
- etc ...

2.3 D'autres approches non paramétriques

2.3.1 La classification floue

Le fondement des méthodes floues est la théorie des ensembles flous introduite par Zadeh [Zad65]. L'appartenance d'un élément à un ensemble n'est plus une valeur vraie ou fausse, mais est donnée par un réel compris entre 0 et 1 appelé degré d'appartenance. Ainsi, un élément peut appartenir à plusieurs ensembles à la fois. Cette notion convient parfaitement à la classification pour modéliser l'appartenance relative d'un point à une classe. On notera ainsi u_{ik} le degré d'appartenance du point x_i à la classe C_k.

2.3.1.1 La famille des C-moyennes

La famille des C-moyennes est une classe de méthodes floues très utilisée en classification. On distingue trois types d'algorithmes suivant les contraintes imposées sur les degrés d'appartenance :

- les C-moyennes ("Hard C-Means" ou HCM)[Dun74] ;

$$\forall i : u_{ik} = 0|1, \sum_{k=1}^{c} u_{ik} = 1 \tag{2.18}$$

– les C-moyennes floues ("Fuzzy C-Means" ou FCM) [Bez87] ;

$$\forall i : u_{ik} \in [0,1], \sum_{k=1}^{c} u_{ik} = 1 \tag{2.19}$$

– les C-moyennes possibilistes ("Possibilistic C-means" ou PCM) [KK93].

$$\forall i : u_{ik} \in [0,1] \tag{2.20}$$

On remarquera ainsi :

$$\boxed{HCM \subset FCM \subset PCM}$$

Ces méthodes sont fondées sur la minimisation d'une fonctionnelle commune :

$$J(U,V;w) = \sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}^{m} d_{ik}^{2} + \sum_{k=1}^{c} w_k \sum_{i=1}^{n} (1-u_{ik})^{m} \tag{2.21}$$

U est la matrice de partition floue d'éléments u_{ik} respectant les contraintes relatives à l'algorithme utilisé. $V = \{V_1, \ldots, V_c\}$ est l'ensemble des prototypes des classes C_k dont la distance aux x_i est noté d_{ik}. $w = \{w_1, \ldots, w_c\}$ est l'ensemble des termes de pénalité de PCM des données atypiques associés à chacune des classes (équivalents à zéro dans le cas de HCM et de FCM). m, appelé coefficient de flou, est un paramètre de l'algorithme contrôlant la quantité de flou dans la partition (m>1).

Le principe de cette famille de méthodes est de minimiser itérativement la fonctionnelle J en alternant une mise à jour de U et de V. La table (2.1) récapitule les équations de mises à jour des éléments de U. On remarque que pour m grand (FCM et PCM), on obtient :

$$\lim_{m\to\infty} u_{ik} = \frac{1}{c}, \qquad \forall i,k$$

En pratique, les degrés d'appartenance sont proches pour m supérieur à 20. A l'inverse, si m tend vers 1, la matrice de partition floue devient une partition stricte (voir table (2.1)). Le choix de m reste ainsi un problème ouvert. Les utilisateurs de cet algorithme choisissent généralement m entre 1 et 2 pour des raisons d'efficacité. De plus, $m = 2$ permet quelques fois de simplifier la mise à jour des prototypes en autorisant une écriture directe de ceux-ci (voir Fuzzy C-Shells).

Dans le cas possibiliste, le degré d'appartenance d'un point à une classe ne dépend pas de celui aux autres classes. Cela est particulièrement intéressant dans le cas de données bruitées car leur degré d'appartenance sera faible pour chacune des classes. Le terme de pénalité w_k peut être considéré comme un seuil sur l'étalement souhaité de la classe. Au

HCM	FCM	PCM
$u_{ik} = \begin{cases} 1 & \text{si } d_{ik} \leq d_{il}, k \neq l \\ 0 & \text{sinon} \end{cases}$	$u_{ik} = \left[\sum_{l=1}^{c} \left(\frac{d_{ik}}{d_{il}}\right)^{\frac{2}{m-1}}\right]^{-1}$	$u_{ik} = \left[1 + \left(\frac{d_{ik}^2}{w_k}\right)^{\frac{1}{m-1}}\right]^{-1}$

TAB. 2.1 – Mise à jour des degrés d'appartenance dans la famille des C-moyennes

delà d'une distance d_{ik} supérieure à w_k, le degré d'appartenance devient rapidement faible voire quasi nul. Il est d'ailleurs très intéressant de remarquer que la formulation du degré d'appartenance dans l'algorithme est très similaire à la fonction de poids du M-estimateur de Cauchy (voir table (3.1) p. 57). Le terme w_k agit ainsi de la même manière que le seuil de Cauchy permettant d'accepter plus ou moins de points en fonction de leur distance au centre de la classe. Ainsi, il est tout à fait possible de dériver de nouvelles versions de PCM en utilisant d'autres M-estimateurs plus performants tels que Huber ou Tuckey (voir chapitre 3). Cependant, du fait de la relaxation de la contrainte de normalisation, l'algorithme PCM peut produire une partition où plusieurs classes sont confondues. On peut y voir un avantage par rapport au problème du choix du nombre de classes. En effet, on peut majorer le nombre de classes recherchées et ne retenir que les c^* classes non confondues à la fin de l'algorithme. Pour limiter le risque d'obtenir nombre de classes confondues, Krishnapuram et Keller recommandent de choisir le coefficient de flou m inférieur à 2 [KJK96]

Dans la suite, on se placera uniquement dans le cadre flou non possibiliste. La fonctionnelle de l'équation (2.21) se réduit à

$$J_f(U,V) = \sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}^m d_{ik}^2 \tag{2.22}$$

La mise à jour des prototypes des classes dépend de la forme des classes recherchées. Quelques algorithmes parmi les plus utilisés vont être présentés par la suite.

2.3.1.1.1 Cas des prototypes-point

L'hypothèse des prototypes-point est qu'un seul point de R^p suffit à représenter la classe C_k. Il n'y a donc aucune modélisation intrinsèque des classes. $V = \{v_1, \ldots, v_c\}$ est l'ensemble des points représentant des classes C_k dont la distance aux x_i est noté d_{ik}.

L'algorithme 7 décrit le fonctionnement de FCM pour les prototypes-points. Le choix de la métrique d_{ik} dépend de la forme des classes recherchées. Pour une métrique euclidienne pondérée, on notera $d_{ik} = ||x_i - v_k||_A = (x_i - v_k)^T A (x_i - v_k)$ où A est une matrice de covariance. Si $A = Id$, les classes recherchées seront des hypersphères. On remarquera que dans ce cas là, l'algorithme HCM peut être interprété comme la version floue de l'algorithme des centres mobiles. Dans le cas contraire, les classes recherchées seront de forme hyperellipsoïdale (voir décomposition de la matrice de covariance page 18).

L'algorithme FCM adopte un fonctionnement analogue à l'algorithme EM. La mise à jour des degrés d'appartenance u_{ik} dans FCM peut être apparentée à l'estimation des probabilités a posteriori dans EM. Dans les deux cas, il s'agit de quantifier la relation qui lie un point à une classe. De la même façon, la mise à jour des points des prototypes dans FCM est comparable à l'étape de maximisation de l'algorithme EM : on ajuste les paramètres du modèle en fonction des résultats de l'étape précédente. La différence entre les deux approches se situe essentiellement au niveau du type de métrique employée. Nous reviendrons sur cette similitude entre les deux approches dans la présentation de l'algorithme FMLE ("Fuzzy Maximum Likelihood Estimation").

Algorithme 7: L'algorithme des C-moyennes floues

Entrée : $\chi = \{x_1, \ldots, x_n\}$ un ensemble de points, c le nombre de classes recherchées, $v^{(0)}$ l'ensemble de prototypes initiaux, d une métrique, m le coefficient de flou, ϵ seuil de stabilité de la partition.

Sortie : Une partition floue U de χ

$t \leftarrow 0$

répéter

{Etape 1 : Mise à jour de la partition floue}

pour *i = 1 à n* **faire**

pour *k = 1 à c* **faire**

$$u_{ik}^{(t)} = \left(\sum_{l=1}^{c} \left(\frac{d_{ik}}{d_{il}}\right)^{\frac{2}{m-1}}\right)^{-1}$$

{Etape 2 : Mise à jour des prototypes des classes}

pour *k = 1 à c* **faire**

$$v_k^{(t+1)} = \frac{\sum_{i=1}^{n} (u_{ik}^{(t)})^m x_i}{\sum_{i=1}^{n} (u_{ik}^{(t)})^m}$$

$t \leftarrow t + 1$;

jusqu'à $||v^{(t+1)} - v^{(t)}|| < \epsilon$;

La modélisation des classes par des prototypes-point impose une uniformité dans la forme des classes recherchées. Une extension naturelle à cette modélisation est de prendre en compte des métriques adaptées selon les classes.

2.3.1.1.2 Modèle de Gustafson-Kessel

Gustafson et Kessel ont proposé d'utiliser des matrices de covariance A_k estimées à partir des degrés d'appartenance u_{ik} [GK79]. Chaque classe est ainsi modélisée par le

couple $V_k = (v_k, A_k)$ La nouvelle fonctionnelle à optimiser s'écrit

$$J_{GK}(U,V) = \sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}^m ||x_i - v_k||_{A_k}^2 \tag{2.23}$$

sous la contrainte $det(A_k) = \rho_k > 0$ qui garantit que A_k soit définie positive. ρ_k qui permet de contrôler le volume de A_k est défini par l'utilisateur.
Cet algorithme est également appelé FCC ("Fuzzy C-Covariance") en raison de la mé-

Algorithme 8: L'algorithme des FCC (Gustafson-Kessel)

Entrée : c le nombre de classes recherchées, $v^{(0)}$ l'ensemble de prototypes initiaux, $A^{(0)}$ l'ensemble des matrices de covariance initiales respectant $det(A_k) = \rho_k$, ρ_k est le volume de la classe C_k, $d_{ik} = ||x_i - v_k||_{A_k}$ la métrique employée, m le coefficient de flou

Sortie : Une partition floue U de χ

$t \leftarrow 0$;

répéter

{Etape 1 : Mise à jour de la partition floue}

pour *i = 1 à n* **faire**

pour *k = 1 à c* **faire**

$$u_{ik} = (\sum_{l=1}^{c}(\frac{d_{ik}}{d_{il}})^{\frac{2}{m-1}})^{-1}$$

{Etape 2 : Mise à jour des prototypes des classes}

pour *k = 1 à c* **faire**

$$v_k = \frac{\sum_{i=1}^{n} u_{ik}^m x_i}{\sum_{i=1}^{n} u_{ik}^m}$$

$$A_k = [\rho_k det(\Sigma_k)]^{1/p}\,\Sigma_k^{-1} \text{ où } \Sigma_k = \frac{\sum_{i=1}^{n} u_{ik}^m (x_i - v_k)(x_i - v_k)^T}{\sum_{i=1}^{n} u_{ik}^m}$$

$t \leftarrow t + 1$;

jusqu'à $||v^{(t+1)} - v^{(t)}|| < \epsilon$;

trique adaptée aux classes impliquant la matrice de covariance.

2.3.1.1.3 Modèles à prototypes linéaires

La métrique utilisée dans la fonctionnelle J_f peut être adaptée à la recherche d'hyperplans H en dimension $p' < p$. Cet algorithme appelé FCV (Fuzzy C-Varieties) permet dans le cas particulier où $p' = 1$ de déterminer des lignes dans un ensemble de points

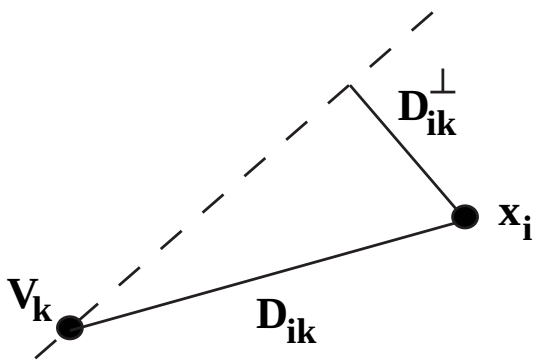


FIG. 2.3 – Métrique pour la recherche des hyperplans

[BCGW81]. Cette variante est appelé FCL (Fuzzy C-lines) [JBW81].
Un hyperplan est défini par combinaison linéaire d'un ensemble $E = \{e_1, \ldots, e_{p'}\}$ de p' vecteurs libres et générateurs (formant une base d'un espace vectoriel de dimension p')

$$y \in H \Longleftrightarrow y = v + \sum_{i=1}^{p'} \alpha_i e_i$$

La métrique considérée est alors la distance $d_{ik}^{\perp}$ de x_i à sa projection orthogonale (voir Fig. (2.3)). On notera ainsi (v_k, V_k) le prototype de la classe C_k où V_k est un ensemble de vecteurs indépendants.

L'ensemble de ces vecteurs $V_k = \{e_{k,1}, \ldots, e_{k,p'}\}$ sont estimés à partir des directions principales de la matrice de covariance floue Σ_k (définie dans l'algorithme 8).

Malheureusement, cet algorithme peut échouer lorsque les hyperplans engendrés par V_k et V_l sont parallèles ou quasiment parallèles. En dimension 1, cela se traduit par une colinéarité ou quasi-colinéarité des vecteurs directeurs (voir Fig. (2.4)).

L'une des corrections possibles à ce problème est d'imposer un critère de compacité sur les classes en combinant la distance orthogonale et la distance euclidienne :

$$\bar{D}_{ik}^2 = \alpha (d_{ik}^{\perp})^2 + (1 - \alpha) d_{ik}^2 \tag{2.24}$$

Le terme α permet de faire un compromis entre la recherche de variétés linéaires et d'ellipsoïdes. On obtient l'algorithme FCE (Fuzzy C-Elliptotypes) [Bez87]. Cet algorithme peut être rendu adaptatif en différenciant les α_k pour chacune des classes. On obtient un nouvel algorithme appelé AFCE (Adaptive Fuzzy C-Elliptotypes) pour lequel il convient d'estimer les α_k [DB92]. En dimension 2, on imagine facilement que le coefficient α est à mettre en relation avec le conditionnement de la matrice de covariance floue Σ_k. Soit $\lambda_{k,1}$ et $\lambda_{k,2}$ les valeurs propres triées par ordre décroissant de Σ_k. Plus cette matrice est mal conditionnée, plus le rapport $\frac{\lambda_{k,2}}{\lambda_{k,1}}$ est proche de 0. Davé a ainsi proposé de remplacer α_k

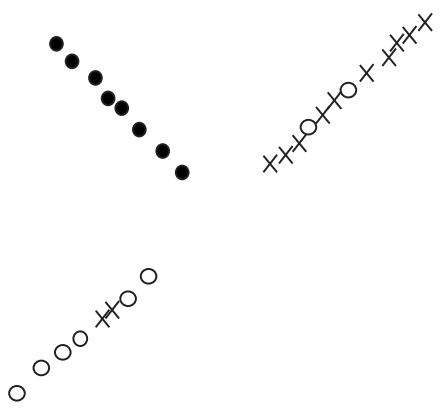


FIG. 2.4 – Cas d'erreur des FCV

par

$$\alpha_k = 1 - \frac{\lambda_{k,2}}{\lambda_{k,1}}$$

Dans le cas sus-cité, la distance $\bar{D}^2_{ik}$ est proche à $(d^{\perp}_{ik})^2$. A l'opposé, si Σ_k est un multiple de la matrice identité, le rapport $\frac{\lambda_{k,2}}{\lambda_{k,1}}$ tend vers 1. On a ainsi $\bar{D}^2_{ik}$ proche de la distance euclidienne. Pour une généralisation en dimension quelconque, le lecteur intéressé pourra se reporter à [PD97].

Modèles à prototypes quadriques

Les modèles présentés jusque là ne permettent pas de rechercher des classes circulaires ou hyperellipses. Cela est possible en choisissant pour prototypes des quadriques définies par le triplet $\mathcal{Q} = (A, b, c)$. L'équation générale d'une quadrique est définie par

$$x^T A x + b^T x + c = 0 \tag{2.25}$$

où A est une matrice symétrique $p \times p$, b un vecteur de dimension p et c un scalaire. En restreignant A à la matrice identité et $b = 0$, on obtient un modèle d'hypersphère

$$d^2_{ik} = (||x_i - v_k|| - r_k)^2$$

de centre v_k et de rayon r_k correspondant à l'algorithme des FCS (Fuzzy C-Shells) [Dav92]. En minimisant la fonctionnelle J_f avec cette distance, on obtient les conditions nécessaires suivantes :

$$\sum_{i=1}^{n} u^m_{ik} \left[1 - \frac{r_k}{||x_i - v_k||}\right](x_i - v_k) = 0 \tag{2.26}$$

$$\sum_{i=1}^{n} u^m_{ik}(||x_i - v_k|| - r_k) = 0 \tag{2.27}$$

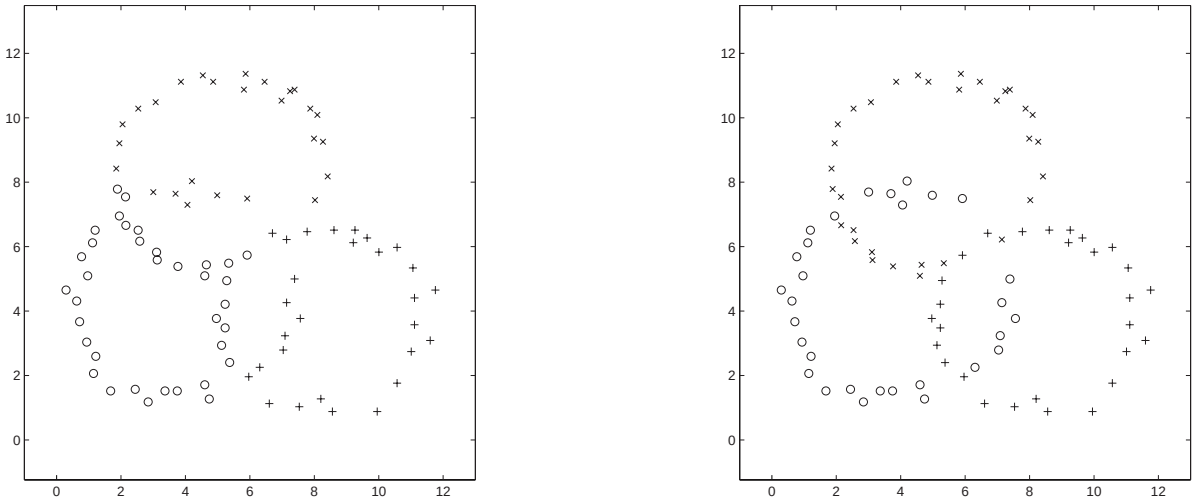


FIG. 2.5 – Un exemple de classification par FCM (à gauche) et FCS (à droite)

Ce système de deux équations non linéaires ne permet pas d'exprimer directement les valeurs de r_k et v_k. On a donc recours à une méthode de résolution numérique du type Newton-Raphson. La figure (2.5) présente un exemple d'application à trois classes de FCM et des FCS.
Ici encore, il est possible d'imposer une métrique $||x_i - v_k||_{A_k}$ propre à chacune des classes. La distance d_{ik}^2 considérée est alors

$$d_{ik}^2 = [||x_i - v_k||_{A_k} - 1]^2 \tag{2.28}$$

Ce modèle, appelé AFCS (Adaptive Fuzzy C-Shells), a été proposé par Davé et Bhaswan [DB92]. Les conditions nécessaires issues de la minimisation de J_f sont alors

$$\sum_{i=1}^{n} u_{ik}^m \left[\frac{d_{ik}}{||x_i - v_k||_{A_k}}\right](x_i - v_k) = 0 \tag{2.29}$$

$$\sum_{i=1}^{n} u_{ik}^m \left[\frac{d_{ik}}{||x_i - v_k||_{A_k}}\right](x_i - v_k)(x_i - v_k)^T = 0 \tag{2.30}$$

La résolution de ce système implique ici également l'usage de méthodes numériques.

Il existe bien d'autres modèles, tant il est aisé de construire une mesure de distance adaptée au contexte. Le lecteur intéressé pourra se référer à un ouvrage de référence [BKKP99].

2.3.1.2 Maximisation de la vraisemblance par une méthode floue

L'algorithme des C-moyennes floues peut être adapté à la décomposition de mélanges gaussiens suivant le principe du maximum de vraisemblance. Cet algorithme appelé FMLE (Fuzzy Maximum Likelihood Estimation) est initialement dû à Bezdek [BD75] puis repris par Gath et Geva [GG89].

La fonctionnelle à minimiser est le terme général de FCM pour $m = 2$

$$J_f(U, V) = \sum_{k=1}^{c} \sum_{i=1}^{n} u_{ik}^2 d_{ik}^2$$

La métrique employée devient de nature probabiliste. Comme dans le cadre de l'algorithme EM, les classes sont représentées par le triplet (π_k, μ_k, Σ_k). La distance des points aux classes est :

$$d_{ik}^2 = \frac{|\Sigma_k|^{1/2}}{\pi_k} exp\left[\frac{(x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k)}{2}\right] \tag{2.31}$$

Cette équation est, à un facteur multiplicatif près, l'équivalent de l'inverse du terme de mélange $(P(C_k)P(x|C_k))$ dans le cas d'une fonction de densité normale. Les degrés d'appartenance sont alors des estimations des probabilités a posteriori :

$$\hat{P}(C_k|x_i) = u_{ik} = \left[\sum_{l=1}^{c} \left(\frac{d_{ik}}{d_{il}}\right)^2\right]^{-1} \tag{2.32}$$

Les équations pour la mise à jour des (π_k, μ_k, Σ_k) sont identiques à celles obtenues dans le cas de l'algorithme EM (voir algorithme 9).

On remarquera que les degrés d'appartenance u_{ik} ne sont pas à la puissance dans les équations (2.34) et (2.33). Ceci permet de dire que l'algorithme FCM est une approximation de l'algorithme FMLE [BD75]. Ainsi, les résultats obtenus par FMLE sont meilleurs que ceux obtenus par FCM surtout dans le cas où les classes sont de forme hyperellipsoïdale. Dans le cas contraire, la différence n'est pas significative [BKKP99]. Certains auteurs préconisent d'initialiser les $\hat{P}(C_k|x_i)$ à partir de la matrice de partition finale obtenue par l'algorithme FCM.

A la fin de l'algorithme de classification et quelque soit la métrique employée, on dispose d'une matrice de partition floue U. Dans un but décisionnel, on désire parfois connaître une affectation stricte des points aux classes. Cela s'effectue en transformant la partition floue en partition stricte. La règle de décision consiste généralement à choisir pour l'élément x_i, la classe C_k dont le degré d'appartenance u_{ik} est le plus fort. Cette règle est analogue au critère du Maximum A Posteriori (MAP) employé dans le cadre

Algorithme 9: L'algorithme FMLE (Fuzzy Maximum Likelihood Estimation)

Entrée : $\chi = \{x_1, \ldots, x_n\}$ un ensemble de points, c le nombre de classes recherchées, Une partition floue $U^{(0)}$, ϵ seuil de convergence

Sortie : Une partition floue U de χ

$t \leftarrow 0$

répéter

{Etape 1 : Mise à jour des prototypes}

pour *k = 1 à c* **faire**

{La moyenne}

$$\hat{\mu}_k^{(t)} = \frac{\sum_{i=1}^{n} u_{ik}^{(t)} x_i}{\sum_{i=1}^{n} u_{ik}^{(t)}} \tag{2.33}$$

{La matrice de covariance}

$$\hat{\Sigma}_k^{(t)} = \frac{\sum_{i=1}^{n} u_{ik}^{(t)} (x_i - \hat{\mu}_k^{(t)})(x_i - \hat{\mu}_k^{(t)})^T}{\sum_{i=1}^{n} u_{ik}^{(t)}} \tag{2.34}$$

{La probabilité à priori}

$$\hat{\pi}_k^{(t)} = \frac{1}{n} \sum_{i=1}^{n} u_{ik}^{(t)} \tag{2.35}$$

{Etape 2 : Mise à jour de la matrice de partition}

pour *i = 1 à n* **faire**

pour *k = 1 à c* **faire**

$$u_{ik}^{(t+1)} = \left[\sum_{l=1}^{c} \left(\frac{d_{ik}}{d_{il}} \right)^2 \right]^{-1}$$

avec

$$d_{ik}^2 = \frac{|\hat{\Sigma}_k^{(t)}|^{1/2}}{\hat{\pi}_k^{(t)}} exp \left[\frac{(x_i - \hat{\mu}_k^{(t)})^T (\hat{\Sigma}_k^{(t)})^{-1} (x_i - \hat{\mu}_k^{(t)})}{2} \right]$$

$t \leftarrow t + 1$;

jusqu'à $||U^{(t+1)} - U^{(t)}|| < \epsilon$;

probabiliste. Dans certains cas, on peut refuser l'affectation lorsque la prédominance du degré d'appartenance à une classe n'est pas marquée. Cela se produit dans les zones où les classes se chevauchent. On parle alors de rejet en ambiguïté.

2.3.2 Classification hiérarchique

Contrairement aux méthodes de classification par partition, les algorithmes de classification hiérarchique ne produisent pas une seule partition mais une hiérarchie de partitions emboîtées. Il existe deux grandes catégories d'algorithmes hiérarchiques : les méthodes ascendantes et les méthodes descendantes. Dans les méthodes ascendantes ou agglomératives, la partition initiale contient autant de classes que de points ($c = n$). A chaque étape, on cherche un couple (C_a, C_b) de classes candidates à la fusion qui maximise (resp. minimise) une certaine mesure de similarité (resp. de dissimilarité). On réitère ce processus jusqu'à n'obtenir qu'une classe contenant tous les éléments.

La figure 2.6 illustre cette hiérarchie de partitions sous une forme appelée dendogramme. Le schéma d'un algorithme de classification hiérarchique ascendante est décrit par l'algorithme (10).

Algorithme 10: Le schéma d'un algorithme de classification hiérarchique ascendante

Données : $\chi = \{x_1, \ldots, x_n\}$, d$(C_a, C_b)$ une mesure de similarité ou de dissimilarité entre les classes C_a et C_b

Sortie : Une hiérarchie de partitions $\{P_0, \ldots, P_{n-1}\}$ de χ

$P_0 = \{\{x_1\}, \ldots, \{x_n\}\}$

pour *i = 1 à n-1* **faire**

  pour chaque *couple* (C_a, C_b) *de* P_{i-1} **faire**

    Choisir $C_{opt} = (C_a, C_b)$ tel que $\begin{cases} d(C_a, C_b) \text{ est maximal si d est une similarité;} \\ d(C_a, C_b) \text{ est minimal si d est une dissimilarité.} \end{cases}$

  $C_{fusion} = C_a \cup C_b$

  $P_i = (P_{i-1} - \{C_a, C_b\}) \cup \{C_{fusion}\}$

A la fin de cet algorithme, on coupe la hiérarchie à un certain niveau de détail ce qui permet de définir le nombre de classes produites. Concernant ce choix, Banfield et Raftery ont proposé une heuristique appelée "AWE" basée sur la vraisemblance classifiante [BR93].

Il existe de nombreuses manières de définir une mesure de similarité ou de dissimilarité. Parmi les plus courantes, citons :

– Critère du lien minimum ("Single-link") :

$$d(C_a, C_b) = min\{d(x, y) \text{ avec } x \in C_a \text{ et } y \in C_b\}$$

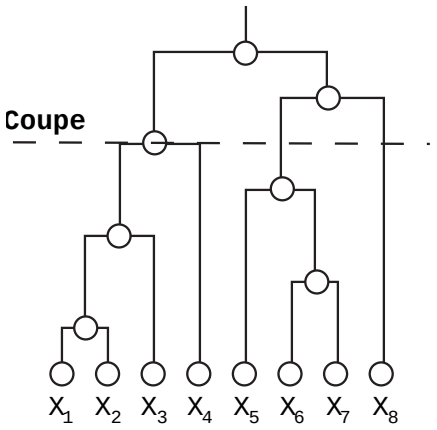


FIG. 2.6 – Dendogramme d'une hiérarchie d'une partition

– Critère du lien complet ("Complete-link") :

$$d(C_a, C_b) = max\{d(x, y) \text{ avec } x \in C_a \text{ et } y \in C_b\}$$

– Critère de la distance moyenne ("Average-link") :

$$d(C_a, C_b) = \frac{\sum_{i=1}^{n_a} \sum_{j=1}^{n_b} d(x_i, y_j)}{n_a * n_b} \text{ avec } x_i \in C_a \text{ et } y_j \in C_b$$

où n_a et n_b désignent respectivement les effectifs des classes C_a et C_b

– Critère de Ward [War63] :

$$d(C_a, C_b) = \frac{n_a * n_b}{n_a + n_b} ||\mu_a - \mu_b||^2 \text{ avec } \mu_a \text{ et } \mu_b \text{ moyennes de } C_a \text{ et } C_b$$

et n_a et n_b leurs effectifs respectifs. On peut montrer que ce critère se résume à minimiser l'augmentation de l'inertie intra-classes.

L'un des avantages de ce type d'algorithme est de fournir, via le dendogramme, une interprétation naturelle du comportement de l'algorithme. A l'opposé, on est généralement confronté à une grande complexité en temps et surtout en espace. En effet, la création de la partition initiale avec un élément par classe nécessite de calculer et de stocker les distances entre tous les couples de points. Cette complexité quadratique peut s'avérer critique pour des jeux de données de grande taille. Récemment, divers travaux ont été menées pour réduire la taille de la partition initiale [Pos99]. Dans ce but, il est également possible d'utiliser un algorithme de classification par partition pour lequel on aura demandé un grand nombre de classes (sur-partitionnement). Cette solution revient ainsi à créer une partition initiale de groupements "évidents".

Les méthodes de classification hiérarchique descendante souffrent des mêmes problèmes d'initialisation. Il convient théoriquement de considérer toutes les partitions possibles en

2 classes, soit $2^{n-1}-1$. Là encore, on a besoin de règles de partitionnement plus simples. Lorsque les données sont totalement supervisées, les arbres de décision peuvent être vus comme un exemple de classification hiérarchique descendante [BFOS84] [Qui93].

2.3.3 Méthodes d'estimation non paramétriques

Dans les méthodes d'estimation de densité par des modèles de mélange, on fait des hypothèses sur la forme analytique des vraisemblances conditionnelles de X. On suppose, par exemple, que les individus appartenant à une classe C_k sont distribués selon une loi normale. Se basant sur un ensemble d'observations, il s'agissait alors d'estimer les paramètres du modèle supposé. La classification était donc vue comme un problème d'identification des paramètres. La démarche non paramétrique ne fait aucune hypothèse sur les distributions a priori des observations. Cette approche convient particulièrement au cas où ces distributions sont inconnues ou multimodales.

Deux techniques sont alors possibles : l'estimation non-paramétrique de la fonction de densité, l'estimation non-paramétrique de la fonction de classement. Dans ce qui suit, seule la première approche sera décrite. La seconde approche inclut les méthodes de classification par arbres de décision et les méthodes neuronales. Les deux méthodes d'estimation de densité présentées ici sont les noyaux de Parzen et les k plus proches voisins.

2.3.3.1 Principe de l'estimation de densité non paramétrique

On suppose ici que l'espace des observations est munie d'une distribution de probabilités définie par une densité f mesurable suivant la mesure de Lebesgue. Soit R une région de l'espace des observations, on peut définir la probabilité d'observer x dans R et son volume par

$$P(R) = \int_R f(x)dx$$

$$V(R) = \int_R dx$$

La densité moyenne de la région R s'écrit $f(R) = \frac{P(R)}{V(R)}$. Dans la mesure où l'on souhaite obtenir la densité ponctuelle d'un point particulier x, on effectue un passage à la limite :

$$f(x) = \lim_{x \in R, V(R) \to 0} f(R) \tag{2.36}$$

Si l'on souhaite estimer $f(x)$ à partir d'un ensemble de n observations, il semble naturel d'utiliser une approche fréquentiste :

$$\hat{P}(R) = \frac{\text{Nombre de points dans R}}{n}$$

Uniforme	$\Phi(y) = \begin{cases} 1/2 & \text{si } y \leq 1 \\ 0 & \text{sinon} \end{cases}$
Linéaire	$\Phi(y) = \begin{cases} 1 - \lvert y \rvert & \text{si } \lvert y \rvert \leq 1 \\ 0 & \text{sinon} \end{cases}$
Gaussien	$\Phi(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}$
Exponentiel	$\Phi(y) = \frac{1}{2} e^{-\lvert y \rvert}$
Epanechnikov	$\Phi(y) = \begin{cases} \frac{3}{4} \frac{(1-\frac{y^2}{5})}{\sqrt{5}} & \lvert y \rvert < \sqrt{5} \\ 0 & \text{sinon} \end{cases}$

TAB. 2.2 – Les principaux noyaux utilisés dans l'estimation par la méthode de Parzen

Cet estimateur permet de définir un estimateur $\hat{f}(R) = \hat{P}(R)/V(R)$ de $f(R)$ non biaisé. D'après l'équation (2.36), il semble raisonnable de poser :

$$\hat{f}(x) = \lim_{x \in R, V(R) \leftarrow 0} \hat{f}(R)$$

Nous allons appliquer les principes décrits ici dans le cadre des noyaux de Parzen et des K plus proches voisins.

2.3.3.1.1 Estimation de la densité par les noyaux de Parzen

La méthode des noyaux de Parzen est fondée sur l'utilisation de fonctions d'influence pour estimer la probabilité $P(R)$ [Par62]. On suppose ainsi que chaque point de données x_i émet une "impulsion" autour de lui dont la forme Φ reste à définir. La somme de ces impulsions au point x permet d'évaluer la densité en ce point. Cela peut s'écrire comme :

$$\hat{f}(x) = \frac{1}{V(R)} \left(\frac{1}{n} \sum_{i=1}^{n} \Phi\left(\frac{||x_i - x||}{h} \right) \right) \tag{2.37}$$

La fonction noyau Φ est paramètrée par la taille h de la fenêtre de lissage. Plus h est élevé, plus grand est le nombre de points considérés dans l'estimation. Cette valeur de h est à relier avec la forme du voisinage choisi. La densité résultante estimée dépend fortement de la taille de la fenêtre de lissage [Dui76]. Pour h trop petit, la courbe de la densité estimée est chaotique laissant apparaître de nombreux modes factices. A l'inverse, la densité estimée est lissée et certains modes réels de f disparaissent. Le choix de la fenêtre de lissage peut être effectuée de manière adaptative en fonction des données observées [VK00].

Il existe de nombreux autres noyaux possibles sous réserve de satisfaire les conditions des densités de probabilité. Les noyaux les plus courants sont récapitulés dans la table 2.2.

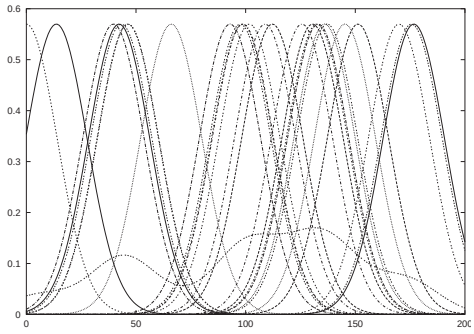


FIG. 2.7 – Estimation non-paramétrique de la densité à l'aide des noyaux de Parzen

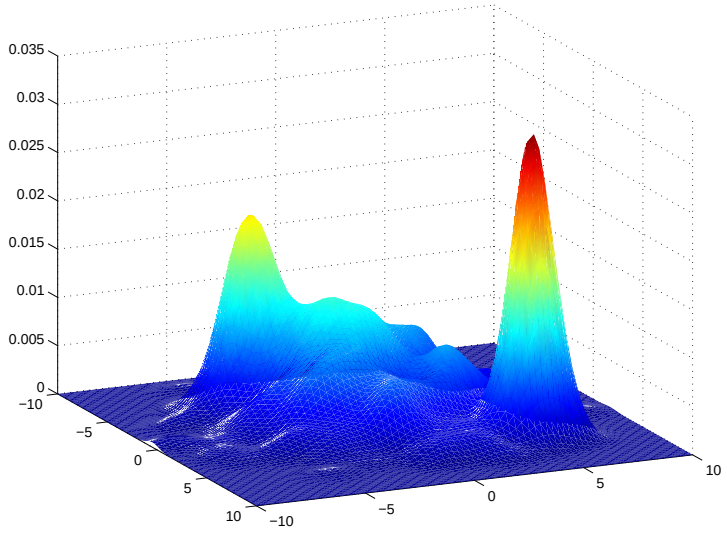


FIG. 2.8 – Exemple d'estimation par la méthode des noyaux de Parzen

La figure (2.8) illustre un exemple d'estimation par la méthode des noyaux de Parzen pour le mélange à trois composantes normales :

$$\mu_1 = \begin{pmatrix} -4 \\ -4 \end{pmatrix}, \Sigma_1 = Id \quad \mu_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_2 = 3 * Id \quad \mu_3 = \begin{pmatrix} 5 \\ 5 \end{pmatrix}, \Sigma_3 = 0.5 * Id$$

2.3.3.1.2 Estimation de la densité par les k plus proches voisins

Dans l'approche des k plus proches voisins, on s'affranchit du choix du paramètre de lissage. Le volume de la région R varie de manière à contenir au moins k points, k étant un paramètre de la méthode. Ainsi, plus V(R) est important, plus la densité estimée sera faible :

$$\hat{f}(x) = \frac{k}{NV_x(R)} \tag{2.38}$$

Dans [Fuk90], on démontre que l'estimateur obtenu est consistent sans biais. On choisit généralement des volumes réguliers centrés en x : hypercubes, hypersphères. Ceux-ci sont déterminés par le type de voisinage considéré (distance de Minkowski, distance euclidienne, etc...).

2.4 Les problèmes en classification

Dans les deux parties précédentes, certains algorithmes ont été présentés en mettant de côté quelques difficultés potentielles. Cette partie va être l'occasion de les aborder.

2.4.1 Choix des paramètres des algorithmes

Dans les algorithmes de classification par partition, le choix du nombre de classes est généralement laissé à l'utilisateur. On peut se demander "Combien existe-t-il de classes dans le mélange ?", mais également "Existe-t-il une structure en classes ?".
Sur la figure (2.9), les données ont été produites par un tirage uniforme sur $[0,1]^2$ et sont visibles à deux niveaux d'échelle différents. Visuellement, on a l'impression qu'il existe 1 (à droite) ou plusieurs classes (à gauche) suivant le niveau de résolution auquel on se trouve.

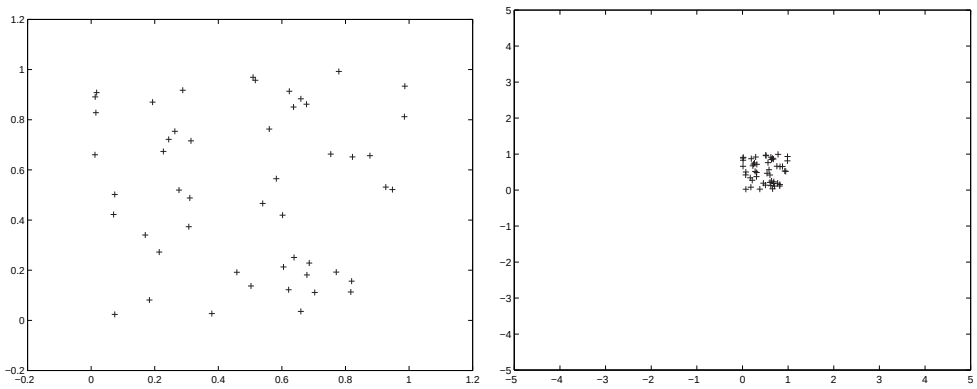


FIG. 2.9 – Tirage uniforme de 50 points dans $[0,1]^2$: Existe t'il une structure ? - une classe ?

Dans le célèbre jeu de données Iris [Fis36], on recherche généralement 3 classes selon l'étiquetage donné par Fisher. Sur la partie gauche de la figure (2.10), on distingue deux classes bien séparées dans la projection de Sammon de ce jeu de données. Sur la partie droite, on peut voir les trois classes données par Fisher (Iris Versicolor, Iris Virginica, Iris Setosa). Deux de ces trois classes sont faiblement mélangées et il semble difficile d'en

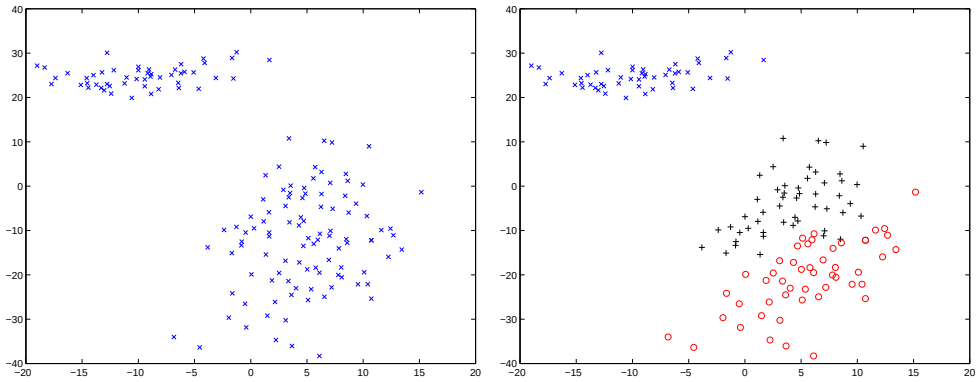


FIG. 2.10 – Projection de Sammon du jeu de données Iris - Les classes selon Fisher

distinguer trois sans connaître les étiquettes.

Il existe plusieurs approches pour déterminer le nombre de composantes d'un mélange. Lorsqu'on suppose le nombre de classes comme paramètre libre, le principe du maximum de vraisemblance conduit généralement à sélectionner un grand nombre de classes car la vraisemblance est une fonction croissante du nombre de composantes.

L'idée des critères d'information est de pénaliser la vraisemblance $L(\mathcal{M})$ par un terme dépendant de la complexité du modèle :

- Akaike [Aka74] :

$$AIC(\mathcal{M}) = -2L(\mathcal{M}) + 2n_p(\mathcal{M})$$

  où $n_p(\mathcal{M})$ désigne le nombre de paramètres libres du modèle $\mathcal{M}$
- "Bayesian Information Criterion" [Sch78] :

$$BIC(\mathcal{M}) = -2L(\mathcal{M}) + \log(n)n_p(\mathcal{M})$$

- Jouzel et al. [JOEM00] :

$$IC(\mathcal{M}) = -2L(\mathcal{M}) + N^{\beta}\log(\log n)n_p(\mathcal{M})$$

  ou $\beta \in]0, 1[$.
- "Information Complexity Criterion" [Boz90] :

$$ICOMP(\mathcal{M}) = -2L(\mathcal{M}) + \frac{n_p(\mathcal{M})}{2}\log(\frac{trF^{-1}}{n_p(\mathcal{M})}) - \frac{1}{2}\log(|F^{-1}|)$$

  où F est la matrice d'information de Fisher du modèle. En pratique, ce critère est difficile à utiliser car il nécessite d'approximer F qui dépend du paramétrage de $\mathcal{M}$ [BG98].

Dans le cas des c composantes gaussiennes non contraintes, le nombre de paramètres libres à estimer est

$$n_p(\{\pi_k, \mu_k, \Sigma_k\}, k = 1, \ldots, c) = \underbrace{c-1}_{\pi_k} + \underbrace{c*p}_{\mu_k} + \underbrace{c*\frac{(p+1)p}{2}}_{\Sigma_k} = \frac{c(p+1)(p+2)}{2} - 1 \quad (2.39)$$

Le critère "Normalized Entropy Criterion" (NEC) utilise l'entropie de la matrice de partition floue pour déterminer le modèle qui produit les classes les plus séparées [Bie97].L'entropie $E(\mathcal{M})$ de la partition U produite sera d'autant plus faible que les classes seront séparées[7]. Le critère NEC s'écrit :

$$\begin{aligned} NEC(\mathcal{M}) &= \frac{E(\mathcal{M})}{L(\mathcal{M}) - L_1(\mathcal{M})} \\ &= \frac{-\sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik} \log u_{ik}}{L(\mathcal{M}) - L_1(\mathcal{M})} \end{aligned} \quad (2.40)$$

où $L(\mathcal{M})$ désigne la vraisemblance classifiante floue et $L_1(\mathcal{M})$ la vraisemblance dans le cas d'une seule composante gaussienne [BG98]. Le lecteur interessé pourra se reporter à la thèse de C. Biernacki [Bie97] ou à l'ouvrage de Bezdek et al. pour d'autres critères de validité d'une partition floue [BKKP99].

Le choix du nombre de classes est effectué en prenant celui dont le critère associé est minimal pour un intervalle d'exploration fixé ($1 \leq c \leq c_{max}$). Le nombre maximal de classes recherchées est choisi parmi plusieurs heuristiques possibles :

- Nombre minimum de points pour l'estimation dans le cas gaussien :

$$c_{max} = \frac{2n}{(p+1)(p+2)}$$

- Critère de XEMgaus [Bie99] :

$$c_{max} = \lceil n^{0.3} \rceil$$

- Bozdogan [Boz90] :

$$c_{max} = \left(\frac{n}{logn}\right)^{\frac{1}{3}}$$

L'une de ces valeurs peut être utilisée pour initialiser l'algorithme PCM. Rappelons que pour cet algorithme, il est possible de démarrer avec un grand nombre de classes puis fusionner, à la fin, les classes confondues.

2.4.2 Méthodes d'initialisation de la partition

Les méthodes itératives de classification par partition sont généralement fondées sur l'optimisation d'une fonctionnelle. Malheureusement, ces fonctions sont hautement non

[7] Pour une partition stricte 0-1, l'entropie est nulle.

linéaires et possèdent de nombreux minima locaux (voir vraisemblance, FCM). De plus, même si cela était possible, l'exploration exhaustive de l'espace des configurations est combinatoirement prohibitive. On a donc recours à des stratégies sous-optimales ou à des heuristiques sujettes aux minima locaux. Les résultats obtenus dépendent bien évidemment de la configuration initiale choisie. Le choix des paramètres initiaux est un problème vaste de la recherche opérationnelle et de nombreuses méthodes sont possibles dans le cas de la classification :

- Tirage aléatoire des centres des classes parmi les points ;
- Utilisation d'une méthode de classification hiérarchique ; On coupe la hiérarchie suivant le nombre de classes désiré et on l'estime les paramètres des classes à partir des sous-ensembles de la partition.
- Utilisation d'un algorithme de classe inférieure dans la famille des C-moyennes ;

  HCM -> FCM, FCM ->PCM
- etc ...

Pour contourner le problème, certains algorithmes intègrent une part d'aléatoire dans le processus d'optimisation. C'est le cas de l'algorithme SEM qui simule une partition par tirage aléatoire selon les distributions a posteriori (voir algorithme 4 page 26). Dans [DBE99], Demiriz et al. minimisent la fonctionnelle d'une variante semi-supervisée des FCM par un algorithme génétique, une heuristique fondée sur des modèles stochastiques.

2.4.3 Dimensionnalité des données

Paradoxalement, l'augmentation du nombre de caractéristiques n'améliore pas systématiquement la qualité de l'apprentissage. En effet, on est confronté au phénomène de Hughes appelé également "the curse of dimensionality" ou “peaking phenomenon” [Bel61]. Ce problème, commun aux algorithmes d'apprentissage, provient du fait que la quantité des données (n) nécessaire pour approximer (apprendre), avec une précision fixée, un concept en dimension p croît exponentiellement en fonction de p . Le lecteur intéressé par ce problème peut se référer à l'exemple de Trunk [Tru79] repris dans [JDM00].

Il existe d'autres arguments en faveur de la réduction de la dimension :

- La complexité des algorithmes augmente lorsque la dimension p croit. Par exemple, la multiplication matricielle s'effectue en $\mathcal{O}(p^{\log_2(7)})$ par la méthode de Strassen [Str69].
- Certains attributs sont séparément pertinents, mais le gain est faible lorsqu'ils sont combinés. Certains attributs peuvent être corrélés (Ex. : Cylindrée d'une voiture et son prix).
- Lorsque la dimension augmente, le nombre de paramètres libres du modèle augmente. Pour une taille des données fixée, le nombre de points impliqués dans ces estimations est insuffisant et la précision de l'estimateur décroît. Il en résulte une

plus faible capacité en généralisation.

2.4.4 Valeurs manquantes

A la fin de l'acquisition, certains attributs d'un ou plusieurs x_i peuvent être manquants. Cela sera par exemple le cas pour un formulaire de sondage indûment rempli. Différentes techniques sont possibles [LR87] [Sar98] pour y remédier :

- Refaire l'acquisition lorsque cela est possible ;
- Effectuer un tirage aléatoire de la valeur manquante
- "Listwise data deletion" : suppression de l'entité x_i ayant au moins une valeur manquante ;
- "Pairwise data deletion" : on utilise les informations connues de l'entité x_i seulement lorsque ses valeurs manquantes ne sont pas utilisées dans un calcul. Dans le cas contraire, l'entité est ignorée ;
- Imputation moyenne : remplacement de la valeur manquante par la moyenne de l'attribut correspondant ;
- Imputation "hot-deck" : remplacement de la valeur manquante par celle de l'exemple le plus proche ;
- Estimation des valeurs manquantes par EM.
- Estimation des valeurs manquantes par régression.

2.4.5 Données aberrantes

Les données aberrantes sont un problème récurrent en classification, pour tout processus d'analyse de données. Nous allons illustrer la pertinence de s'intéresser à ce problème en évoquant quelques-unes des causes possibles de présence des données aberrantes.

Le problème des données aberrantes est particulièrement sensible dans les sciences expérimentales. Par exemple, pour une étude comparative du comportement à la corrosion de différents alliages, on peut choisir d'analyser un ensemble d'échantillons soumis à des conditions expérimentales les plus proches possibles. Cependant, il existe toujours un biais car tous échantillons fabriqués ne peuvent être rigoureusement identiques. Pour cet exemple, une donnée aberrante peut être constituée d'un échantillon présentant un défaut de préparation (fêlure, impuretés) et qui aboutit à un comportement anormal à la corrosion. L'analyse des résultats obtenus peut ainsi être faussée. Ce genre de données aberrantes peuvent être qualifiées d'aberrantes par nature.

Par opposition, on peut définir les données aberrantes artificielles comme celles qui sont

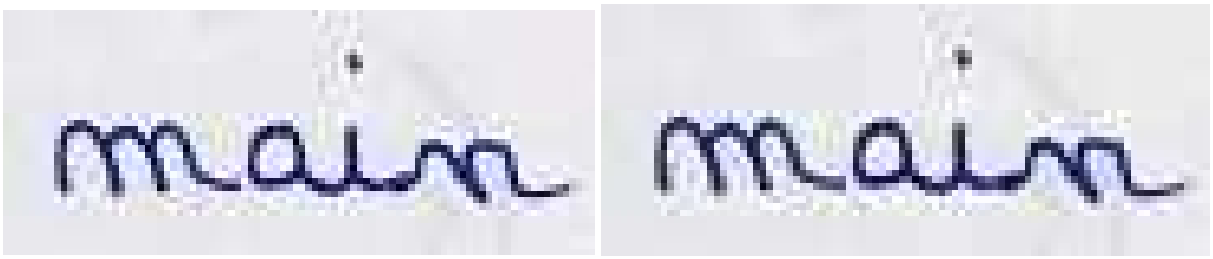


FIG. 2.11 – Robustesse au stockage : A gauche, l'image originale ; à droite, l'image comprimée par la méthode JPEG à 50%

perturbées à l'issue d'une acquisition ou d'une transformation a posteriori des données. Le cas le plus évident est la panne partielle ou totale d'un capteur (dans ce dernier cas, la valeur de l'attribut sera manquante). Cela peut arriver pour des capteurs auto-alimentés (batteries vides), pour une caméra ayant une rayure sur l'objectif (défaut chronique), un appareil de mesure peu tolérant aux conditions de température, etc...

D'autres causes possibles de la dégradation des données sont liées au transport et au stockage des données. Concernant le transport de l'information, on peut citer l'exemple de la transmission d'un signal analogique ou numérique par les ondes hertziennes (téléphones portables, radio, television) qui ont conduit à l'utilisation d'amplificateur ou de code correcteur (Ex. : code correcteur de Hamming). Dans ce cas, on parlera plutôt en termes de de données bruitées (rapport signal/bruit) que de données aberrantes. Néanmoins, la conception d'une méthode de classification robuste nécessite de prendre en compte ce type de perturbation. Les problèmes liées au stockage des données sont de même nature. Il convient en général d'effectuer un compromis entre espace et qualité de l'information restituée. Un illustration parfaite de ce problème pour le traitement des données audiovisuelles est la compression des images et de la musique. On est souvent conduit à choisir des taux de compression ne dégradant pas trop visuellement ou auditivement les données. Cette qualité perceptuelle est déconnectée du traitement machine effectué en aval. L'exemple le plus célèbre est certainement l'encodage JPEG des images et des vidéos (Motion-JPEG). Cet algorithme de compression est adapté aux images naturelles mais possède un comportement ératique sur les images possédant un fort contraste. La figure (2.11) nous montre les fameux artefacts carrés du JPEG sur un exemple de texte scannerisé. On peut ainsi voir que le traitement par bloc transforme des pixels d'intensité faible en intensité moyenne et réciproquement d'intensité forte en intensité moyenne. Il est parfois possible de restaurer les images dégradées par des techniques de filtrage (Ex. : filtrage de Wiener pour le deblurring). Concernant les données sonores, les techniques de compression actuelles (comme le MP3) sont basées sur des modèles psycho-acoustiques

humains. Pour gagner de la place, certaines fréquences inaudibles trop hautes ou trop basses sont supprimées. Il en résulte une altération de la qualité sonore qui peut être perçue différemment selon les personnes. Pour la machine et suivant l'application visée, cette dégradation est plus ou moins dommageable (Ex. : localisation ultra-sonore).

Nous venons de voir que les données aberrantes et le bruit peuvent provenir de causes multiples. Bien évidemment, il est souhaitable de les éliminer dès que possible. Cela peut être effectué de manière simple lorsque ces données sont vraiment très atypiques. Dans le cas contraire, la génération des caractéristiques, la réduction de dimension (Extraction/Sélection) et la classification bien sûr peuvent s'en trouver affectées (Ex. : Calcul de la matrice de covariance pour l'ACP). Ce dernier point va faire l'objet d'une grande partie du prochain chapitre et justifie l'emploi de techniques robustes en classification.

Chapitre 3

Robutesse et Semi-supervision en classification

Dans le chapitre précédent, nous avons décrit diverses méthodes de classification automatique. Ici, nous allons aborder les deux thématiques principales de cette étude, à savoir, la robustesse et la semi-supervision en classification. Certains algorithmes présentés auparavant seront étendus dans une de ces deux directions. La première partie est consacrée à la justification et à l'emploi de techniques robustes en classification. Après avoir défini ce qu'est la robustesse, nous nous intéresserons plus particulièrement aux M-estimateurs et à leur cadre d'application. Par la suite, nous décrirons trois approches distinctes de la classification robuste à travers la présentation de différents algorithmes.
La seconde partie constitue un état de l'art des méthodes de classification semi-supervisée. Comme pour la classification automatique, nous distinguerons l'approche paramétrique des autres approches.

3.1 Robustesse et classification

Le terme "robustesse" a été introduit par Box [Box79]. Au sens général, un système est qualifié de robuste lorsqu'il se comporte de façon monotone, même s'il est soumis à des conditions non-standard. Pour des données relativement éloignées du cas idéal, la réponse du système ne doit que peu ou pas être perturbée, et l'échec total ne doit avoir lieu que dans des conditions suffisamment extrêmes. Bien entendu, la quantité de distorsion apportée au système reste à définir. Ce point sera discuté ultérieurement. Même dans les cas les plus simples, on fait souvent des hypothèses implicites ou explicites sur les observations. Dans la version standard de l'algorithme des centres mobiles, on recherche implicitement des classes sphériques de par la métrique utilisée (euclidienne non pondérée). Généralement, l'hypothèse d'indépendance entre les observations est également faite. Toutes ces hypothèses, même si l'on sait qu'elles sont plus ou moins exactes, n'ont pour but que de simplifier le traitement d'un problème.

En 1981, Huber parle de "robustesse distributionnelle"[8] pour qualifier l'écart entre la distribution de probabilités réelle et le modèle supposé (le plus souvent gaussien) [Hub81]. Les données aberrantes sont qualifiées de distorsions faibles dans la mesure où elles sont en faible proportion.

3.1.1 Préambule

3.1.1.1 Propriétés d'un estimateur

Lorsqu'on parle d'estimation, on est amené à définir les propriétés d'un estimateur. Pour illustrer notre propos, prenons le cas simple de l'estimateur T de l'espérance mathématique $\theta = E(X)$ à partir d'un échantillon $\chi = x_1, \ldots, x_n$ de taille n où X est une variable aléatoire distribuée selon une loi normale. On notera $\hat{\theta} = T_n(x_1, \ldots, x_n)$, la valeur estimée de θ par T_n.

Par exemple, on connaît diverses formules pour estimer l'espérance :

- La moyenne arithmétique : $\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- La médiane : $\hat{\theta} =$ med $\{x_i\}$
- Le mode : $\hat{\theta} =$ mod $\{x_i\}$ (valeur la plus courante)

Ces estimateurs ne sont pas équivalents. Pour les comparer, on dispose de diverses notions :

- **Le biais**
  On dira qu'un estimateur est non biaisé si son espérance est égale à la vraie valeur θ :
  $$E(T_n) = \theta$$
  L'estimateur ne sera qu'asymptotiquement sans biais si :
  $$\lim_{n \to +\infty} E(T_n) = \theta$$
  Ainsi, un estimateur biaisé approche la valeur vraie avec un décalage systématique. Par exemple, la moyenne arithmétique est un estimateur non biaisé de l'espérance.
- **La convergence**
  Un estimateur est convergent si lorsque la taille de l'échantillon tend vers l'infini, la valeur obtenue est la valeur vraie :
  $$\exists n_0 \text{ tel que } \forall n > n_0 \; \forall \epsilon, \gamma \; P(|T_n(x_1, \ldots, x_n) - \theta| < \epsilon) > 1 - \gamma$$

[8] Traduction libre de "Distributional robustness"

– **La précision**
Un estimateur est précis ou efficace s'il donne des résultats proches d'un échantillon à l'autre. Cette précision est déterminée par la variance de l'estimateur :

$$E((T_n - E(T_n))^2)$$

– **La complexité**
La complexité d'un estimateur désigne le temps (théorique) de calcul en fonction de la taille de l'échantillon. Pour calculer la médiane, il est nécessaire d'effectuer un tri ce qui porte la complexité de l'estimateur à $\mathcal{O}(n\log(n))$.
– **la robustesse**
Un estimateur est robuste s'il est peu sensible aux données aberrantes. On utilise en particulier la notion de point de rupture pour mesurer la robustesse d'un estimateur.

3.1.1.2 Point de rupture

La notion de point de rupture ("breakdown point" en anglais) a été introduite par Donoho et Huber [DH83] pour quantifier la robustesse d'un estimateur. De manière intuitive, le point de rupture d'un estimateur est la quantité de points aberrants dans un jeu de données mettant en défaut cet estimateur. Plus formellement, cela peut s'écrire pour un échantillon $\chi = \{x_1, \ldots, x_n\}$ de taille n et un estimateur T :

$$\epsilon_n^*(T, \chi) = \min\{\epsilon_n^+(T, \chi), \epsilon_n^-(T, \chi)\}$$

avec

$$\epsilon_n^+(T, \chi) = \min_m\{\frac{m}{n}; \sup_{\chi'} T(\chi') = +\infty\}$$

et

$$\epsilon_n^-(T, \chi) = \min_m\{\frac{m}{n}; \inf_{\chi'} T(\chi') = 0\}.$$

χ' correspond à χ pour lequel m observations ont été remplacées par des valeurs arbitraires (cela signifie aussi grandes ou aussi petites que l'on veut). Les ϵ_n^+ et ϵ_n^- sont respectivement appelés le point de rupture d'explosion et d'implosion de l'estimateur.

Dans les formules précédentes, on remarquera que la notion de rupture est associée à la taille n de l'échantillon. Dans certains cas, on préférera la version asymptotique de la définition :

$$\epsilon^*(T) = \lim_{n->+\infty} \epsilon_n^*(T, \chi)$$

Cette mesure va nous permettre de comparer la robustesse de deux estimateurs précédents de l'espérance : la moyenne arithmétique (MA) et la médiane (MED). Soit $\chi = \{x_1, \ldots, x_n\}$ un ensemble de n nombres réels.

Théorème 1 *La moyenne arithmétique a un point de rupture égal à 0.*

Preuve : Commençons par prouver $\epsilon_n^+(MA, \chi) = \frac{1}{n}$

$$MA(\chi) = \frac{x_1 + \ldots + x_n}{n}$$

Soit χ_i' l'élément qui a remplacé x_i avec x_i' arbitrairement grand : $x_i' \to \infty$.
On a alors

$$MA(\chi') = \frac{x_1 + \ldots + x_i' + \ldots + x_n}{n} \to \infty$$

avec $\chi' = \{x_1, \ldots, x_i', \ldots, x_n\}$.

De manière évidente, on a également $\epsilon_n^-(MA, \chi) = \frac{1}{n}$ car il suffit de prendre

$$x_i' = -(x_1 + \ldots + x_{i-1} + x_{i+1} + \ldots + x_n)$$

pour obtenir $MA(\chi') = 0$.
En passant à la limite, on obtient

$$\epsilon^*(MA) = \lim_{n->+\infty} \epsilon_n^*(MA, \chi) = \lim_{n->+\infty} \frac{1}{n} = 0$$

•
Ce théorème nous prouve qu'il suffit d'un seul point contaminé pour mettre cet estimateur en défaut.

Théorème 2 *La médiane a un point de rupture égal à* $\frac{1}{2}$.

Preuve : La preuve est quasi-immédiate pour $\epsilon_n^+(MED, \chi)$ et $\epsilon_n^-(MED, \chi)$. Soit $\delta = max(\chi)$ pour que $MED(\chi')$ soit supérieur à δ, il faut au moins $\lfloor \frac{n}{2} \rfloor + 1$ données supérieures à δ. On a donc

$$\epsilon_n^+(MED, \chi) = \frac{\lfloor \frac{n}{2} \rfloor + 1}{n}$$

De même pour avoir, $MED(\chi') = 0$ il suffit d'avoir $\lfloor \frac{n}{2} \rfloor + 1$ valeurs de χ' à zéros. Ainsi,

$$\epsilon_n^-(MED, \chi) = \frac{\lfloor \frac{n}{2} \rfloor + 1}{n}$$

Par passage à la limite, on obtient

$$\epsilon^*(MED) = \lim_{n->+\infty} \frac{\lfloor \frac{n}{2} \rfloor + 1}{n} = \frac{1}{2}$$

•

3.1.2 Les M-estimateurs

3.1.2.1 Cadre général

Dans cette partie, nous présentons le cas général de l'estimation robuste à l'aide des M-estimateurs [Hub81]. On souhaite estimer le paramétrage $\theta = (\theta_1, \ldots, \theta_m)$ d'un modèle à partir d'un échantillon $\chi = \{x_1, \ldots, x_n\}$. Soit e_i l'écart entre la donnée observée x_i et la prévision de cette donnée $\hat{x}(i, \theta)$ par le modèle :

$$e_i(\theta) = x_i - \hat{x}(i, \theta)$$

Au sens des moindres carrés (Least-Squares), on cherche à minimiser l'erreur $E_{LS}(\theta)$ commise par le modèle qui s'écrit :

$$E_{LS}(\theta) = \sum_{i=1}^{n} e_i(\theta)^2$$

Cette solution n'est pas très satisfaisante car l'erreur E_{LS} est très sensible aux données aberrantes (le point de rupture est ici de 0). Nous verrons par la suite que ce modèle correspond à affecter un poids équivalent à tous les e_i.

La solution proposée par Huber [Hub81] consiste à minimiser une fonction d'erreur plus générale :

$$E_\rho(\theta) = \sum_{i=1}^{n} \rho(e_i(\theta)) \tag{3.1}$$

où $\rho(y) = log(J(y)^{-1})$ désigne la contribution à l'erreur de l'écart y et J la distribution de ces écarts. A ce propos, Lippmann disait :

> "Everyone believes in the normal law of errors : the mathematicians, because they think it is an experimental fact ; and the experimenters, because they suppose it is a theorem of mathematics."

Malheureusement, cela n'est ni un théorème des mathématiques ni un fait expérimental couramment observé. La fonction ρ est continue, symétrique ayant un minimum en zéro (généralement égal à zéro). La minimisation de l'équation (3.1) conduit à résoudre le système différentiel à m inconnues suivant :

$$\sum_{i=1}^{n} \psi(e_i(\theta)) \frac{\partial e_i(\theta)}{\partial \theta_m} = 0 \qquad \text{pour } k = 1, \ldots, m \tag{3.2}$$

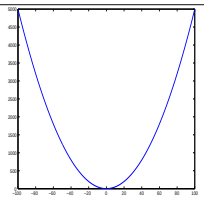
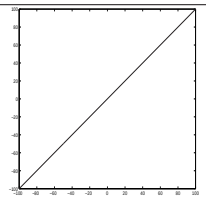
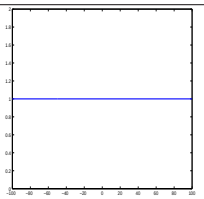
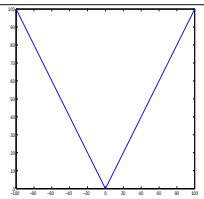
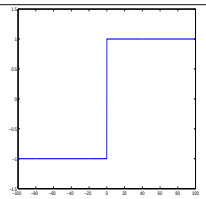
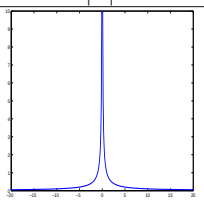
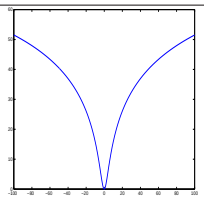
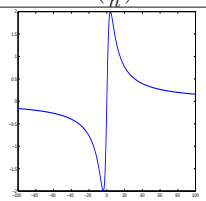
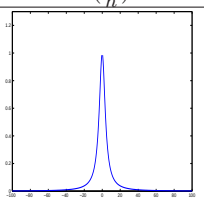
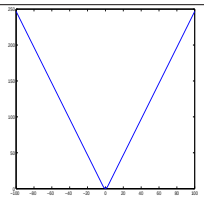
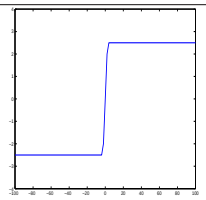
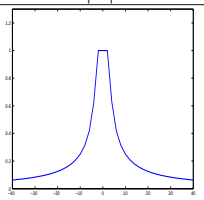
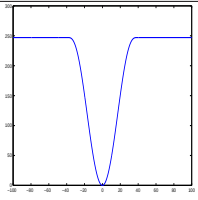
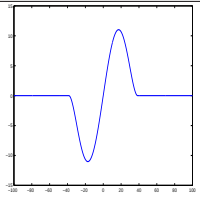
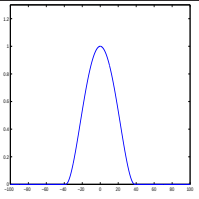
où $\psi(r) = \frac{d\rho(r)}{dr}$ est appelée fonction d'influence. Ce système n'a pas de solution générale et il convient de l'étudier selon la fonction ρ. Par la suite, nous noterons $w(r) = \frac{\psi(r)}{r}$ la fonction de poids associée au résidu r, écart entre l'observation et la prédiction.

3.1.2.2 Quelques estimateurs

Nous allons maintenant présenter quelques M-estimateurs parmi les plus utilisés. La table (3.1) résume leurs fonctions ρ, ψ, w associées. Certains d'entre eux nécessitent des paramètres que nous noterons par la suite h et S. h désigne un facteur d'échelle alors que S peut être assimilé à un écart-type.

- **Modèle de Legendre :**
  Le modèle de Legendre correspond à la méthode des moindres carrés. Les écarts sont supposées être distribués selon la loi normale ($J(e) = \exp^{-\frac{e^2}{\sqrt{2}}}$). La fonction d'influence croît linéairement en fonction de l'erreur : les points aberrants vont donc avoir une grande influence et la minimisation de E_{LS} reviendra à minimiser l'erreur sur les points aberrants.
- **La médiane :**
  La médiane également appelée estimateur L_1 évite ce problème en bornant la fonction d'influence ($\psi = -1$ ou 1). Minimiser la fonction de coût précédemment décrite revient à calculer le résidu médian.
- **Le modèle de Cauchy/Lorentz :**
  Ce modèle permet une décroissance plus lente contrôlée par le paramètre c. Pour $h = 2.3849$, l'estimateur prend en compte 95% des points de la loi $\mathcal{N}(0, 1)$ [Zha95].
- **Le modèle de Huber :**
  Le modèle de Huber est souvent considéré comme l'un des meilleurs estimateurs car il associe une décroissance forte (quadratique) à une zone de confiance contrôlée par le paramètre h. Ce paramètre permet également de moduler la vitesse de décroissance : plus h est faible, plus la zone de confiance est étroite et plus la décroissance est rapide. D'après [Zha95], la discontinuité de la dérivée seconde de ρ peut être la cause d'une éventuelle instabilité en pratique. Ainsi, Rey a proposé une modification du modèle de Huber garantissant cette continuité [Rey83].
- **Le modèle de Tuckey :**
  Le modèle de Tuckey est plus complexe que le modèle de Huber. La constante de Huber est ici remplacée par hS, appelée également point de rejet. Au delà de ce seuil, les données sont simplement éliminées du processus d'estimation. Les paramètres S et h peuvent être assimilés respectivement à un l'écart-type et son facteur multiplicateur. Pour une loi normale centrée réduite, 95% des réalisations auront un poids non nul pour $h = 4.6851$ et $S = 1$.

La figure (3.1) représente les évolutions des fonctions de poids des estimateurs de Cauchy, Huber et Tuckey en fonction de leurs paramètres. Pour le modèle de Tuckey, la courbe présentée correspond à $S = 1$. Pour chacun de ces estimateurs, augmenter le seuil h permet de prendre en compte plus de données dans l'estimation. Ainsi, ce qui est gagné en robustesse et perdu en précision. Il convient, dès lors, de choisir précautionneusement ces seuils de manière à faire un bon compromis entre précision et robustesse.

Modèle	$\rho(x)$	$\psi(x)$	$w(x)$
Legendre	x^2	$2x$	2
			
Médiane	$\lvert x\rvert$	$sgn(x)$	$\frac{1}{\lvert x\rvert}$
			
Cauchy Lorentz	$\frac{h^2}{2}\log(1+(\frac{x}{h})^2)$	$\frac{x}{1+(\frac{x}{h})^2}$	$\frac{1}{1+(\frac{x}{h})^2}$
			
Huber	$\frac{x^2}{2}$ si $\lvert x\rvert \leq h$ $h\lvert x\rvert - \frac{c^2}{2}$ sinon	x $h\ sgn(x)$	1 $\frac{h}{\lvert x\rvert}$
			
Tuckey	$\frac{1}{6}((1-(\frac{x}{hS})^2)^3)$ si $\lvert x\rvert \leq hS$ $\frac{1}{6}$ sinon	$x(1-(\frac{x}{hS})^2)^2$ 0	$(1-(\frac{x}{hS})^2)^2$ 0
			

TAB. 3.1 – Les M-estimateurs les plus courants

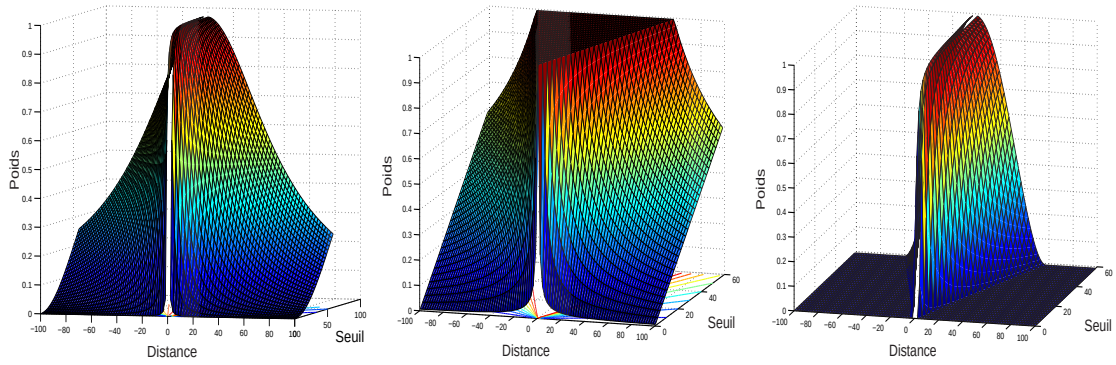


FIG. 3.1 – Les fonctions de poids de Cauchy, Huber, Tuckey en fonction de leurs paramètres

Nous allons maintenant présenter trois approches robustes de la classification dont une repose sur l'utilisation d'un M-estimateur. Auparavant, nous allons rappeler la nécessité de prendre en compte les points aberrants.

3.1.3 Méthodes robustes en classification

3.1.3.1 Le problème des points aberrants en classification

Les points aberrants constituent une difficulté majeure en classification et en particulier pour les méthodes itératives de recherche de partition. On fait ici l'hypothèse que les données n'ont pas été filtrées.

Dans les algorithmes de classification par optimisation d'une fonctionnelle (Ex. : EM, FCM), la minimisation ou la maximisation est effectuée sous contraintes. Pour les FCM par exemple, on utilise la contrainte de Ruspini [Rus69] :

$$\sum_{k=1}^{c} u_{ik} = 1, \text{ pour } i = 1, \ldots, n \tag{3.3}$$

où u_{ik} désigne le degré d'appartenance de x_i à la classe C_k. Dans les méthodes probabilistes, on impose une contrainte sur les probabilités a posteriori :

$$\sum_{k=1}^{c} P(C_k|x_i) = 1, \text{ pour } i = 1, \ldots, n \tag{3.4}$$

Ces méthodes estiment itérativement les paramètres des classes à partir des points x_i sur la base des pondérations u_{ik} calculées précédemment. Les contraintes (3.3) et (3.4) étant semblables, certains auteurs ne distinguent d'ailleurs les approches floues des approches probabilistes que par la méthode d'obtention de ces valeurs. u_{ik} désignera donc

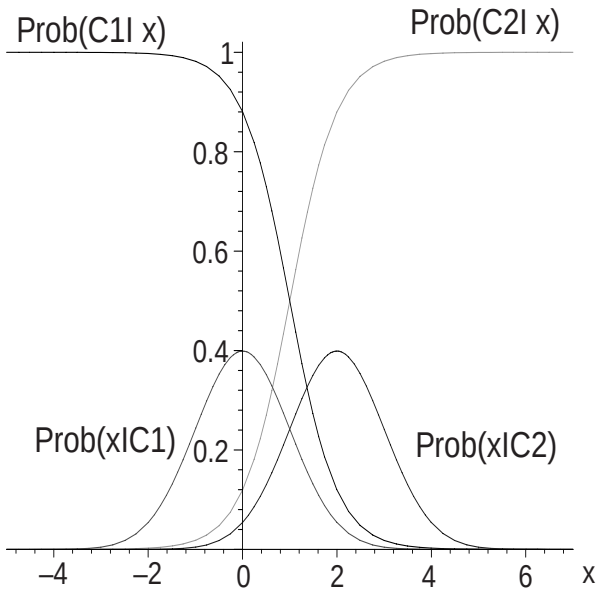


FIG. 3.2 – Les probabilités a posteriori dans le cas de 2 classes gaussiennes

ici invariablement le degré d'appartenance ou la probabilité a posteriori. Pour les points aberrants, les u_{ik} tendent vers des valeurs binaires (0-1) à l'exception de ceux proches des frontières de décision :

- $u_{ik} \sim 1$ si la distance entre la classe C_k et le point aberrant x_i est minimale ;
- $u_{il} \sim 0$ pour les classes $C_l (l \neq k)$

La figure (3.2) montre les probabilités a priori et a posteriori dans le cas de deux classes gaussiennes C1 et C2 mono-dimensionnelles. La figure (3.3) représente les probabilités aposteriori dans le cas des trois classes gaussiennes C1, C2 et C3 bi-dimensionnelles. Les u_{ik} sont égaux sur les frontières de décision. Les poids des points aberrants sont maximaux pour au moins une des classes. Ils comptent ainsi autant que les données "propres" issues de la classe dans l'estimation. Par conséquent, les centres des classes seront attirés par ces données aberrantes. Dans la conception d'un système robuste, on doit donc chercher à limiter l'influence de ces points. Plusieurs approches vont être présentées maintenant.

3.1.3.2 Une méthode de classification robuste par compétition (RCA)

Dans cette partie, la méthode de classification robuste par compétition proposée par Frigui et al. [FK99] est présentée d'une manière non exhaustive. Seuls les aspects relatifs à cette étude seront détaillés. Cette méthode, fondée sur l'algorithme des C-moyennes floues (présenté en section 2.3.1.1), utilise un M-estimateur pour calculer de manière robuste les paramètres des classes. Pour cette méthode, le choix du nombre de classes n'est pas effectué explicitement mais implicitement en fixant des seuils pour la compétition entre

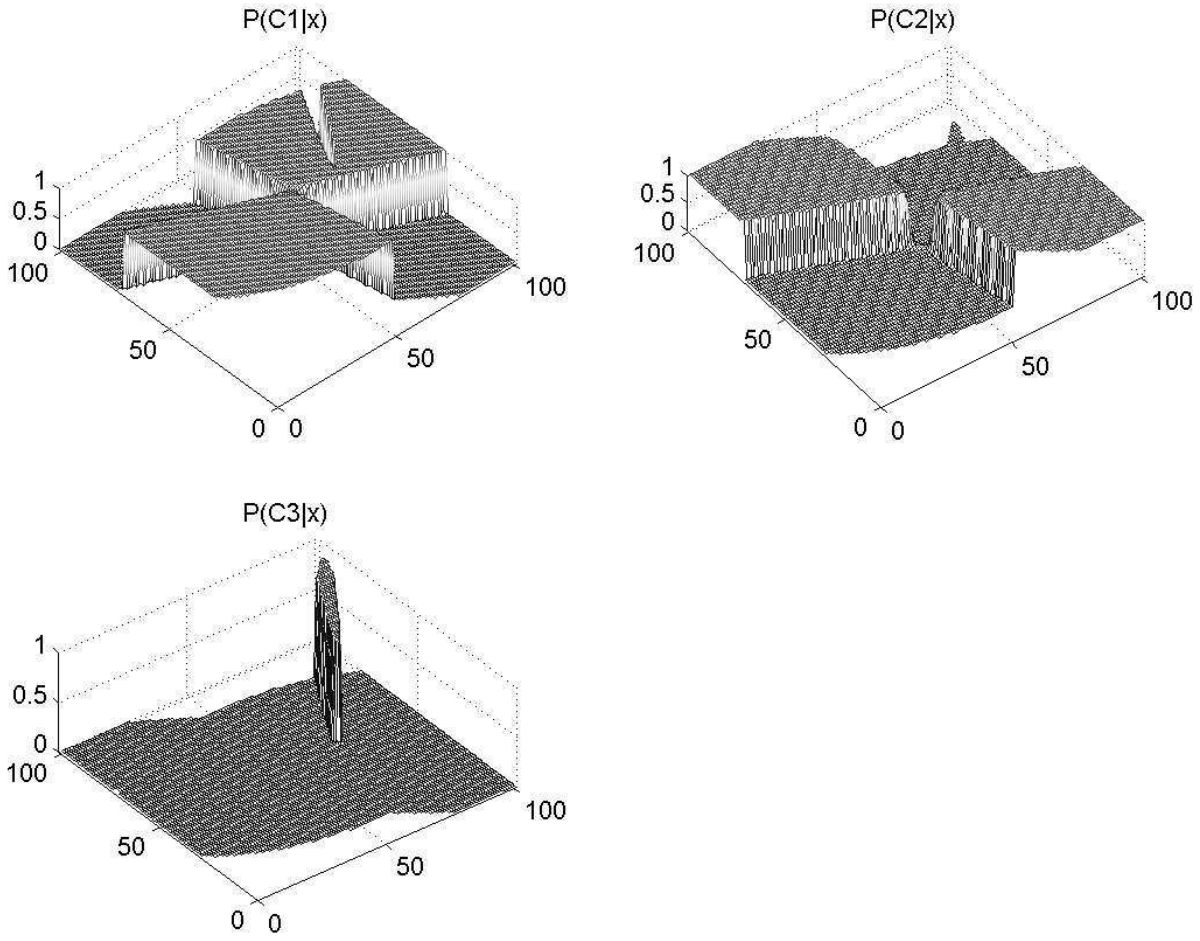


FIG. 3.3 – Les probabilités a posteriori dans le cas de 3 classes gaussiennes. Le poids des points aberrants est proche de 0 ou de 1 à l'exception des frontières de décision)

les classes (Ex. : le nombre minimal de points par classe).

3.1.3.2.1 La modification de la fonctionnelle

Dans le cas des FCM, on rappelle que la fonctionnelle à minimiser est (Eq. (2.22)) :

$$J(U,V) = \sum_{k=1}^{C}\sum_{i=1}^{N} u_{ik}^{m} d_{ik}^{2}$$

où u_{ik} et d_{ik} désignent respectivement le degré d'appartenance et la distance de x_i à la classe C_k. Cette minimisation est effectuée sous les contraintes (2.19). En laissant le nombre de classes comme paramètre libre du système, on aboutit à la solution triviale d'un point par classe ($c = n$). Cette fonctionnelle peut être modifiée par l'introduction d'un terme de régularisation qui prend la forme :

$$J_r(\Theta, P) = \sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}^{2}\rho_k(d_{ik}^{2}) - \alpha\sum_{k=1}^{c}(\sum_{i=1}^{n} w_{ik}u_{ik})^{2} \tag{3.5}$$

sous les mêmes contraintes de minimisation que précédemment en fixant le coefficient de flou m à 2. ρ_k désigne le M-estimateur (et son paramètrage) associé à la k-ième classe et w_{ik} le poids donné par cet estimateur à l'élément x_i. Le second terme représente la somme des carrés des cardinalités robustes qui est minimale pour le cas d'une seule classe regroupant tous les points (Inégalité de Cauchy-Schwartz). Le terme α permet d'effectuer un compromis entre classes compactes et classes de grande taille. La métrique utilisée est la distance euclidienne pondérée :

$$d_{ik}^{2} = ||x_i - v_k||_{A_k}^{2} \tag{3.6}$$

3.1.3.2.2 La mise à jour des degrés d'appartenance et des divers paramètres

La minimisation de (3.5) modifie les formules de mise à jour des degrés d'appartenance qui deviennent :

$$u_{ik} = \frac{1/\rho_k(d_{ik}^{2})}{\sum_{l=1}^{c} 1/\rho_l(d_{il}^{2})} + \frac{\alpha}{\rho_k(d_{ik}^{2})}(N_k - \bar{N}_i) = u_{ik}^{RR} + u_{ik}^{Biais} \tag{3.7}$$

Le premier terme u_{ik}^{RR} est le degré avec lequel la classe C_k partage x_i avec les autres classes. Dans le cadre probabiliste, cette quantité peut s'assimiler à la notion de probabilité a posteriori $P(C_k|x_i)$. Dans le second terme, N_k désigne la cardinalité robuste de la classe C_k et $\bar{N}_i$ représente la moyenne pondérée des cardinalités des classes autour de x_i :

$$N_k = \sum_{i=1}^{n} w_{ik}u_{ik} \tag{3.8}$$

$$\bar{N}_i = \frac{\sum_{k=1}^{c} \frac{1}{\rho_k(d_{ik}^{2})} N_k}{\sum_{k=1}^{c} \frac{1}{\rho_k(d_{ik}^{2})}} \tag{3.9}$$

Lorsque un point x_i appartenant à la classe C_k n'est pas en compétition ($u_{ik} \approx 1$), on obtient $N_k \approx \bar{N}_i$. A l'inverse lorsque x_i est proche de plusieurs classes, il y a compétition entre celles-ci sur la base de leurs cardinalités en fonction de α. Les classes ayant une cardinalité inférieure à un certain seuil sont éliminées et le nombre de classes c est décrémenté.

Le paramètre α peut être déterminé avec une heuristique fondée sur un principe proche du recuit simulé. Lors de l'initialisation de l'algorithme, le nombre de classes est majoré : $c = c_{max}$. Dans les premières itérations, une valeur forte de α permet une forte compétition entre les classes pour faire diminuer rapidement c. Au fur et à mesure que l'algorithme progresse, α décroît de plus en plus lentement jusqu'à la stabilisation de c. Les auteurs proposent d'utiliser la règle suivante :

$$\alpha^{(t)} = \eta^{(t)} \frac{\sum_{k=1}^{c} \sum_{i=1}^{n} (u_{ik}^{(t-1)})^2 \rho_k((d_{ik}^2)^{(t-1)})}{\sum_{k=1}^{c} \left[\sum_{i=1}^{n} w_{ik}^{(t-1)} u_{ik}^{(t-1)}\right]^2} \tag{3.10}$$

où t désigne l'itération courante et $\eta^{(t)}$ est une loi de décroissance en fonction de t. On choisira par exemple :

$$\eta^{(t)} = \begin{cases} \eta^{(0)} e^{-|t_0 - t|/\tau} & \text{si } t > 0 \\ 0 & \text{sinon} \end{cases} \tag{3.11}$$

où $\eta^{(0)}, t_0, \tau$ sont des constantes fixées par l'utilisateur.

La convergence des paramètres des classes est utilisée comme critère d'arrêt de la procédure.

La fonction de poids w_i choisie dans cette méthode est une variante de l'estimateur de Tuckey : le poids associé à un point dont la distance au modèle supérieure à un certain seuil est nul. Cet estimateur s'écrit :

$$w_{ik} = w_i(d_{ik}^2) = \begin{cases} 1 - \frac{d_{ik}^4}{2T_i^2} & \text{si } d_{ik}^2 \in [0, T_i] \\ \frac{[d_{ik}^2 - (T_i + \beta S_i)]^2}{2c^2 S_i^2} & \text{si } d_{ik}^2 \in [T_i, T_i + \beta S_i] \\ 0 & \text{si } d_{ik}^2 > T_i + \beta S_i \end{cases} \tag{3.12}$$

où d_{ik}^2 représente le carré de la distance entre x_i et C_k. La fonction de poids est définie par intervalles (cf. Fig. (3.4)) :

- (A) $[0, T_i]$ - zone de confiance
- (B) $[T_i, T_i + \beta S_i]$ - zone de doute
- (C) $[T_i, +\infty[$ - zone de rejet (points aberrants)

Comme pour l'estimateur de Huber, cette fonction de poids n'est pas continûment dérivable ce qui peut influer sur la stabilité de l'estimation.
Le paramétrage de l'estimateur peut être effectué à l'aide de statistiques locales individuelles aux classes, par exemple :

$$T_k = Med_i(d_{ik}^2) \tag{3.13}$$

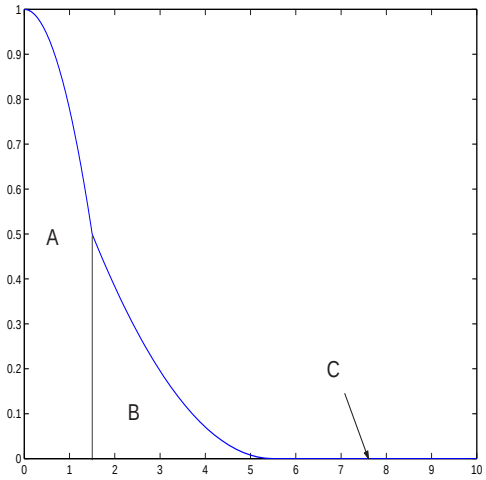


FIG. 3.4 – La fonction de poids de Frigui et al

$$S_k = MAD_i(d_{ik}^2) = Med_i(|d_{ik}^2 - Med_j(d_{jk}^2)|) \tag{3.14}$$

L'écart absolu médian (Median Absolute Deviation) est généralement considéré comme la version robuste de l'écart-type. Il fournit une indication sur l'étalement des mesures autour de la valeur médiane. Le paramètre β est un coefficient multiplicateur de S permettant de contrôler la taille de la zone de doute. Les auteurs de la méthode recommandent de choisir β relativement élevé (entre 4 et 12) ce qui laisse supposer que la quasi-totalité des points sont pris en compte dans l'estimation. Il est également possible d'utiliser une loi de décroissance de β en fonction du nombre d'itérations :

$$\beta^{(t)} = max(\beta_{min}, \beta^{(t-1)} - \Delta\beta) \tag{3.15}$$

où β_{min} est une valeur minimum fixée par l'utilisateur. La décroissance de la fonction de poids est forte (quadratique) sur le premier intervalle (zone de confiance). Le nombre de points ayant un poids élevé (proche de 1) va être relativement faible.

Cette méthode de dérivation du seuil du M-estimateur est à mettre en relation, avec la formule généralement utilisée. Pour avoir l'écart absolu médian égal à l'écart type de la distribution, on introduit un terme de correction du biais correspondant à la médiane de la loi normale réduite (Si $X \sim N(0,1)$, $MAD(X) \approx Med(|X|) \approx 0.6745 = \frac{1}{1.4826}$) :

$$MAD_i(d_{ik}^2) = 1.4826 Med_i(|d_{ik}^2 - Med_j(d_{jk}^2)|)$$

Le choix précédent du paramètre S_k de l'estimateur peut ainsi être reformulé en facteur β_k de l'écart-type :

$$S_k = \beta_k 1.4826 Med_i(|d_{ik}^2 - Med_j(d_{jk}^2)|) \tag{3.16}$$

3.1.3.2.3 La mise à jour des paramètres des classes

La mise à jour des paramètres des classes doit être effectuée en tenant compte des poids associés à chacun des individus. Pour la moyenne et la matrice de covariance, on peut par exemple utiliser les formules :

$$v_k = \frac{\sum_{i=1}^{n} (u_{ik})^2 \; w_{ik} \; x_i}{\sum_{i=1}^{n} (u_{ik})^2 \; w_{ik}} \tag{3.17}$$

$$A_k = \frac{\sum_{i=1}^{n} (u_{ik})^2 \; w_{ik} \; (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{n} (u_{ik})^2 \; w_{ik}}, \tag{3.18}$$

Ainsi, malgré un degré d'appartenance u_{ik} proche de 1, le produit $[(u_{ik})^2 \; w_{ik}]$ sera faible pour les données aberrantes.

La méthode de classification RCA est récapitulée dans l'algorithme 11.

Algorithme 11: RCA : Un exemple d'algorithme utilisant les M-estimateurs

Entrée : Une valeur initiale des prototypes $(v_k, A_k)^{(0)}$, un ensemble d'observations χ, c_{max} le nombre maximal de classes, n_{min} le nombre de points minimal par classe

Sortie : Une suite stationnaire $\{\Theta^{(t)}\}$ d'estimation des paramètres du modèle

$t \leftarrow 0$;

Choisir le nombre maximal de classes $c = c_{max}$;

répéter

- Calculer d_{ik}^2 pour $i = 1, \ldots, n$ et $k = 1, \ldots, c$ (Eq. 3.6);
- Estimer les seuils des M-estimateurs ρ_k pour chaque classe (Eq. 3.13) (Eq. 3.14);
- Mettre à jour les poids w_{ik} pour $i = 1, \ldots, n$ et $k = 1, \ldots, c$ (Eq. 3.12) ;
- Mettre à jour le taux de compétition α (Eq. 3.10);
- Mettre à jour la matrice de partition P (Eq. 3.7);
- Calculer la cardinalité robuste N_k pour $k = 1, \ldots, c$ (Eq. 3.8);
- Supprimer les classes de cardinalité inférieure à n_{min} (Eq. 3.8);
- Mettre à jour le nombre de classes c;
- $t \leftarrow t + 1$;
- Mettre à jour le coefficient multiplicateur β (Eq. 3.15);
- Mettre à jour les prototypes Θ_k des classes (Eq. 3.17) (Eq. 3.18);

jusqu'à *Convergence des paramètres*;

3.1.3.3 FNC : Ajout d'une classe de bruit

Alors que l'algorithme RCA fait diminuer explicitement l'influence des points aberrants par l'utilisation d'un M-estimateur, une autre approche consiste à créer une classe de

bruit et à estimer le degré d'appartenance à cette classe. L'algorithme FNC (Fuzzy Noise Clustering) introduit par Davé modifie l'algorithme des Fuzzy C-Means par l'introduction d'une classe supplémentaire destinée à prendre en compte les données bruitées [Dav91]. On rappelle que dans le contexte de la classification par la méthode FCM, les degrés d'appartenance u_{ik} sont soumis à la contrainte $\sum_{k=1}^{c} u_{ik} = 1$ pour tout élément x_i. L'idée de l'auteur est d'autoriser une classe supplémentaire en relâchant cette contrainte. Si l'on note u_{i*} le degré d'appartenance à cette classe, la nouvelle contrainte devient :

$$u_{i*} + \sum_{k=1}^{c} u_{ik} = 1 \text{ ou de manière équivalente } u_{i*} = 1 - \sum_{k=1}^{c} u_{ik} \tag{3.19}$$

La distance d_{i*} des points à cette classe fictive est un paramètre de l'algorithme commun à tous les points. La fonctionnelle à minimiser devient

$$J_{NC}(U, V; d_{i*}) = \sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}^m d_{ik}^2 + \sum_{i=1}^{n} u_{i*}^m d_{i*}^2 \tag{3.20}$$

La règle de mise à jour des degrés d'appartenance doit être modifiée pour respecter les nouvelles contraintes :

$$u_{ik} = \frac{(1/d_{ik}^2)^{1/m-1}}{\sum_{l=1}^{c}(1/d_{il}^2)^{1/m-1} + (1/d_{i*}^2)^{1/m-1}} \tag{3.21}$$

Dans cette équation, on remarque que le second terme du dénominateur est constant. Ainsi, pour une donnée aberrante, le numérateur et le premier terme du dénominateur tendant vers 0, le degré d'appartenance de ce point à toutes les classes réelles devient quasi-nul. Il résulte que u_{i*} est proche de 1.

Le paramètre d_{i*} fixé par l'utilisateur permet de contrôler la vitesse de décroissance de u_{ik} en fonction de la distance aux classes de données. Pour mieux appréhender son influence, prenons le cas de classes très bien séparées. Sous cette hypothèse, le terme $\sum_{l=1}^{c}(1/d_{il}^2)^{1/m-1}$ tend vers par $(1/d_{ik}^2)^{1/m-1}$. Le degré d'appartenance u_{ik} se comportera donc comme

$$u_{ik} \sim \frac{(1/d_{ik}^2)^{1/m-1}}{(1/d_{ik}^2)^{1/m-1} + (1/d_{i*}^2)^{1/m-1}}$$

La figure (3.5), représentant u_{ik} en fonction de d_{ik} et d_{i*}, confirme un comportement analogue à l'estimateur de Cauchy (voir table (3.1)). d_{i*} joue ainsi un rôle analogue au terme de pénalité w_k dans l'algorithme PCM mais sans toutefois être différencié pour chacune des classes. Cet algorithme peut être utilisé avec diverses métriques comme la distance euclidienne pondérée adaptée aux classes. L'algorithme (12) constitue l'extension FNC de la méthode de Gustafson-Kessel(FCC).

Algorithme 12: L'algorithme FNC (Fuzzy Noise Clustering) pour les classes ellipsoïdales

Entrée : c le nombre de classes recherchées, $v^{(0)}$ l'ensemble de prototypes initiaux, $A^{(0)}$ l'ensemble des matrices de covariance initiales respectant $det(A_k) = \rho_k$, ρ_k est le volume de la classe C_k, $d_{ik} = ||x_i - v_k||_{A_k}$ la métrique employée, d_{i*} la distance fixe des points à la classe de bruit, m le coefficient de flou

Sortie : Une partition floue U de χ

$t \leftarrow 0$;

répéter

{Etape 1 : Mise à jour de la partition floue}

pour *i = 1 à n* **faire**

pour *k = 1 à c* **faire**

$$u_{ik} = \frac{(1/d_{ik}^2)^{1/m-1}}{\sum_{l=1}^{c}(1/d_{il}^2)^{1/m-1} + (1/d_{i*}^2)^{1/m-1}}$$

$$u_{i*} = 1 - \sum_{k=1}^{c} u_{ik}$$

{Etape 2 : Mise à jour des prototypes des classes}

pour *k = 1 à c* **faire**

$$v_k = \frac{\sum_{i=1}^{n}(u_{ik})^m x_i}{\sum_{i=1}^{n}(u_{ik})^m}$$

$$A_k = [\rho_k det(\Sigma_k)]^{1/p}\, \Sigma_k^{-1} \text{ où } \Sigma_k = \frac{\sum_{i=1}^{n} u_{ik}^m (x_i - v_k)(x_i - v_k)^T}{\sum_{i=1}^{n} u_{ik}^m}$$

jusqu'à $||U^{(t+1)} - U^{(t)}|| < \epsilon$;

$t \leftarrow t + 1$;

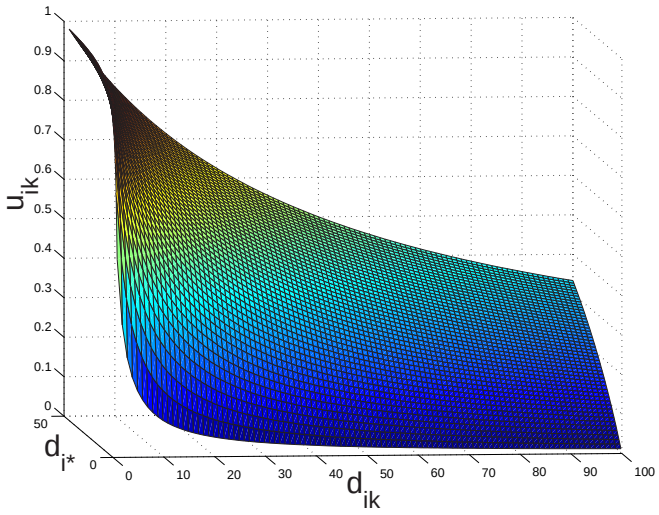


FIG. 3.5 – FCN : le degré d'appartenance en fonction de d_{ik} et d_{i*}

L'approche de la robustesse développée dans cette méthode peut être reproduite pour l'algorithme PCM en effectuant une α-coupe sur les degrés d'appartenance. On met ainsi les u_{ik} à zéro s'ils sont inférieurs à un certain seuil α. Si tous les u_{ik} sont nuls pour un x_i donné, celui-ci sera classé dans la classe de bruit. Sur le principe, cela revient à attribuer, comme pour FNC, un degré d'appartenance à une classe fictive pour chacun des points. Dans [DK97], Davé et Khrishnapuram unifient les algorithmes FNC et PCM en introduisant un degré d'appartenance différencié pour chacun des points dans FCN.

3.1.3.4 Utilisation d'un modèle de bruit : t-distribution

Une autre façon de traiter les points aberrants est de les inclure dans le modèle théorique des classes. Au lieu d'utiliser un simple modèle gaussien, on va considérer des modèles de contamination de données. En effet, le modèle gaussien est peu adapté aux données aberrantes car les valeurs situées à plus de 3 fois l'écart-type de la moyenne ont une probabilité a priori quasi-nulle ($< 0.3\%$). On peut remplacer ainsi le modèle gaussien par une distribution de Student (ou t-distribution). Dans sa forme générale, ce modèle de paramètres μ, Σ, ν s'écrit :

$$f(x|\mu,\Sigma,\nu) = \frac{\Gamma(\frac{\nu+p}{2})|\Sigma|^{-1/2}}{(\pi\nu)^{\frac{p}{2}}\Gamma(\frac{\nu}{2})(1+d^2_{\mathcal{M}}(x;\mu,\Sigma))^{\frac{\nu+p}{2}}} \tag{3.22}$$

où μ, Σ, ν désignent respectivement la moyenne, la matrice de covariance et le nombre de degrés de liberté de la loi. $d_{\mathcal{M}}$ est la distance de Mahalanobis [Mah36]. L'expression de la fonction Gamma est donnée dans l'annexe B. La figure (3.6) permet de comparer la différence entre les queues de distribution de la loi normale et de la loi de Student centrées

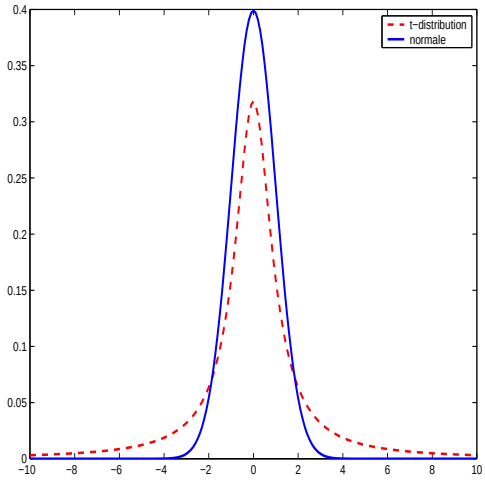


FIG. 3.6 – Comparaison entre la loi normale et la loi de Student

réduites.

Dans la représentation hiérarchique[9] du modèle de Student par une combinaison de lois normales et Gamma, on constate qu'utiliser une t-distribution est équivalent à considérer un modèle de contamination tel que la densité de probabilité est décrite par :

$$(1-\gamma)\mathcal{N}(x|\mu,\Sigma)+\gamma\mathcal{N}(x|\mu,\alpha\Sigma) \tag{3.23}$$

où γ contrôle les proportions entre les deux modes et α l'étalement de la seconde composante. Cette équation peut se réécrire sous la forme :

$$X_i|U_i, Z_{ik}=1 \sim \mathcal{N}(\mu_k, \frac{1}{u_i}\Sigma_k) \tag{3.24}$$

$$U_i|Z_{ik}=1 \sim (\frac{\nu_k}{2}, \frac{\nu_k}{2}) \tag{3.25}$$

où Z_{ik} indique l'appartenance ou non de X_i à la classe C_k. La variable U peut être vue comme une notion de typicalité qui modifie la variance dans la loi normale.

On peut utiliser l'algorithme EM pour estimer les paramètres d'un mélange de c composantes de Student [MP98]. Le vecteur de paramètres à estimer est alors $\Theta = [\pi_1,\ldots,\pi_c,\Theta_1,\ldots,\Theta_c]^T$ avec $\Theta_k=[\mu_k,\Sigma_k,\nu_k]^T$. Comme il a été vu dans le chapitre précédent, les données x_i sont considérées comme incomplètes et sont augmentées par le vecteur d'appartenance stricte $z_i=[z_{i1},\ldots,z_{ic}]^T$. Le vecteur de typicalité $u_i=[u_{i1},\ldots,u_{ic}]^T$ peut également être considéré comme une donnée manquante. La donnée complète y_i s'écrit alors $y_i=(x_i,z_i,u_i)$.

[9] Lorsqu'on souhaite construire une densité multidimensionnelle complexe, un modèle hiérarchique est une recomposition de cette densité par des étapes séparées plus simples[KRT99]

L'étape E-Step consiste à calculer les quantités :

$$\hat{z}_{ik}^{(t)} = \frac{\hat{\pi}_k^{(t)} f(x_i|\hat{\mu}_k^{(t)}, \hat{\Sigma}_k^{(t)}, \hat{\nu}_k^{(t)})}{\sum_{l=1}^{c} \hat{\pi}_l^{(t)} f(x_i|\hat{\mu}_l^{(t)}, \hat{\Sigma}_l^{(t)}, \hat{\nu}_l^{(t)})} \tag{3.26}$$

$$\hat{u}_{ik}^{(t)} = \frac{\hat{\nu}_k^{(t)} + p}{\hat{\nu}_k^{(t)} + d_{\mathcal{M}}(x_i; \hat{\mu}_k^{(t)}, \hat{\Sigma}_k^{(t)})} \tag{3.27}$$

L'étape de maximisation est effectuée grâce aux formules :

$$\hat{\pi}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}}{n} \tag{3.28}$$

$$\hat{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} \hat{u}_{ik}^{(t)} x_i}{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} \hat{u}_{ik}^{(t)}} \tag{3.29}$$

$$\hat{\Sigma}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} \hat{u}_{ik}^{(t)} (x_i - \hat{\mu}_k^{(t+1)})(x_i - \hat{\mu}_k^{(t+1)})^T}{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}} \tag{3.30}$$

Les détails des calculs amenant à ces équations sont dans l'article [PM00]. La mise à jour de ν_k n'est pas disponible dans une formulation directe et $\hat{\nu}_k^{(t+1)}$ doit être calculée numériquement en résolvant (voir Eq. (32) dans [PM00]) :

$$\left\{ -\psi(\frac{\hat{\nu}_k^{(t)}}{2}) + \log(\frac{\hat{\nu}_k^{(t)}}{2}) + 1 + \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} (\log \hat{u}_{ik}^{(t)} - \hat{u}_{ik}^{(t)})}{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}} + \psi\left(\frac{\hat{\nu}_k^{(t)} + p}{2}\right) - \log\left(\frac{\hat{\nu}_k^{(t)} + p}{2}\right) \right\} = 0 \tag{3.31}$$

La convergence de l'algorithme EM est très lente en cas d'estimation conjointe des paramètres μ_k,Σ_k,ν_k. Ainsi, les auteurs proposent de séparer l'estimation des moyennes et matrices de covariance de l'estimation des degrés de liberté en utilisant l'algorithme ECM [MK97]. Dans une première étape, on calcule les μ_k,Σ_k qui maximisent la vraisemblance tout en fixant ν_k. On réitère ensuite l'opération en fixant cette fois à μ_k,Σ_k. L'attrait principal de cette méthode réside dans le fait que le modèle s'ajuste automatiquement aux points aberrants par la détermination du degré de liberté ν_k.

La robustesse de l'algorithme est assurée par l'introduction du terme de typicalité u_{ik}. Dans l'équation (3.27), on constate que pour un degré de liberté fixé, u_{ik} décroît en fonction de la distance de Mahalanobis. Ainsi, les pondérations $\hat{z}_{ik}\hat{u}_{ik}$ utilisées dans l'estimation des moyennes et des matrices de covariance réduisent l'influence des points aberrants. On remarquera également l'absence de $\hat{u}_{ik}$ dans le dénominateur de l'équation (3.30). Celui-ci peut être remplacé par $\sum_{i=1}^{n} \hat{z}_{ik}\hat{u}_{ik}$ pour accélérer la convergence dans le

cas d'une composante [PM00].

Dans [MP00], Mac Lachlan et Peel déterminent les points aberrants par un seuillage de la valeur $\delta_i = \sum_{k=1}^{c} \hat{u}_{ik} \max_l(\hat{z}_{il})$. δ_i est faible si le point x_i est atypique. On retrouve ici le principe d'une α-coupe. A l'inverse de deux premières méthodes, la robustesse est incluse directement dans la modélisation théorique des classes.

Algorithme 13: L'algorithme EM pour les t-distributions

Données : Une valeur initiale des paramètres du modèle $\Theta^{(0)}$, un ensemble d'observations χ, c le nombre de classes

Sortie : Une estimation $\hat{\Theta}$ des paramètres du modèle

$t \leftarrow 0$

Initialisation du modèle $\Theta^{(0)}$

répéter

E-Step (Estimation)

pour *i = 1 à n* **faire**

pour *k = 1 à c* **faire**

$$\hat{z}_{ik}^{(t)} = \frac{\hat{\pi}_k^{(t)} f(x_i|\hat{\mu}_k^{(t)}, \hat{\Sigma}_k^{(t)}, \hat{\nu}_k^{(t)})}{\sum_{l=1}^{c} \hat{\pi}_l^{(t)} f(x_i|\hat{\mu}_l^{(t)}, \hat{\Sigma}_l^{(t)}, \hat{\nu}_l^{(t)})}$$

$$\hat{u}_{ik}^{(t)} = \frac{\hat{\nu}_k^{(t)} + p}{\hat{\nu}_k^{(t)} + d_{\mathcal{M}}(x_i; \hat{\mu}_k^{(t)}, \hat{\Sigma}_k^{(t)})}$$

M-Step (Maximisation)

pour *k = 1 à c* **faire**

$$\hat{\pi}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}}{n}$$

$$\hat{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} \hat{u}_{ik}^{(t)} x_i}{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} \hat{u}_{ik}^{(t)}}$$

$$\hat{\Sigma}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)} \hat{u}_{ik}^{(t)} (x_i - \hat{\mu}_k^{(t+1)})(x_i - \hat{\mu}_k^{(t+1)})^T}{\sum_{i=1}^{n} \hat{z}_{ik}^{(t)}}$$

Résoudre numériquement l'équation (3.31) pour $\hat{\nu}_k^{(t+1)}$

$t \leftarrow t + 1$

jusqu'à *Convergence*;

3.2 Classification Semi-supervisée

Dans cette partie, un état de l'art des méthodes de classification semi-supervisée est présenté. La semi-supervision intervient lorsqu'on dispose à la fois d'un ensemble de données étiquetées χ^d et non étiquetées χ^u. On note $n_d = card(\chi^d)$ et $n_u = card(\chi^u)$. Généralement, les données non supervisées sont disponibles en grand nombre car peu coûteuses à produire. Au contraire, les données supervisées qui nécessitent l'expertise humaine sont plus rares mais également plus riches d'information. On distingue deux approches duales de la semi-supervision suivant le point de départ où l'on se place :

- Utilisation de la donnée supervisée pour aider un algorithme de classification automatique ;
- Utilisation d'un grand volume de données non étiquetés pour améliorer un apprentissage supervisé.

L'approche développée dans cette étude correspond essentiellement au premier point. En effet, on peut considérer que l'on ne dispose originellement d'aucune donnée étiquetée. Dans ce cas, on est contraint d'utiliser un algorithme de classification non supervisée et l'information supervisée est intégrée au fur et à mesure de sa disponibilité.

3.2.1 L'intérêt de la semi-supervision en classification

Dans cette partie, l'intérêt de la semi-supervision est rappelée à travers des exemples. Les points supervisés seront matérialisés par des symboles • ou des ∘ dans les figures.

Le premier exemple considéré contient deux classes séparées et quelques points isolés. La classification obtenue est le résultat de l'algorithme FCM présenté précédemment (voir Fig. (3.7)). L'estimation des paramètres des classes est perturbée par les données aberrantes. L'algorithme privilégie des classes bien séparées en dépit de la densité des deux composantes. La version standard des FCM réalise une minimisation pondérée au sens des moindres carrés. En déplaçant l'ensemble des points aberrants sur la droite, on arrivera toujours à mettre en défaut l'algorithme car l'influence des données n'est pas bornée ($\sim d_{ik}^2$).

Si l'on ajoute quelques points supervisés (voir Fig. 3.8), on contraint suffisamment le processus d'optimisation pour éviter un extremum local très éloigné de l'optimum global pour autant que les points supervisés soient pertinents (sélection non biaisée des données étiquetées).

La semi-supervision trouve également son intérêt dans des problèmes de type OU-Exclusif. C'est d'ailleurs, sur ce type de données que Pedrycz a utilisé et justifié pour

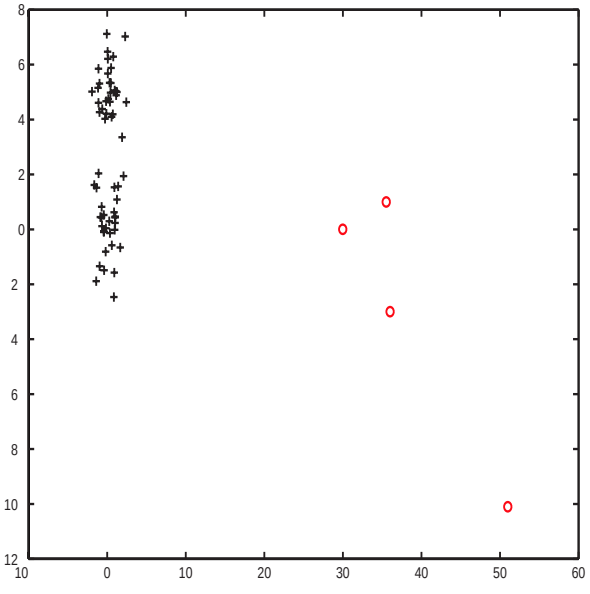


FIG. 3.7 – FCM : Sensibilité aux données aberrantes (2 classes demandées)

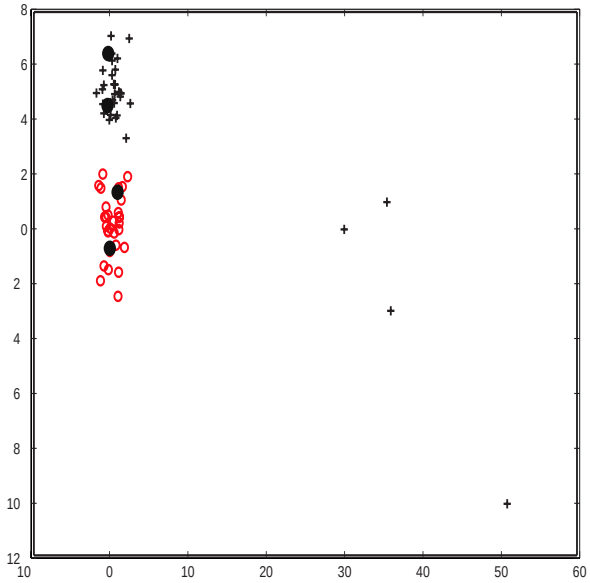


FIG. 3.8 – Correction par la supervision

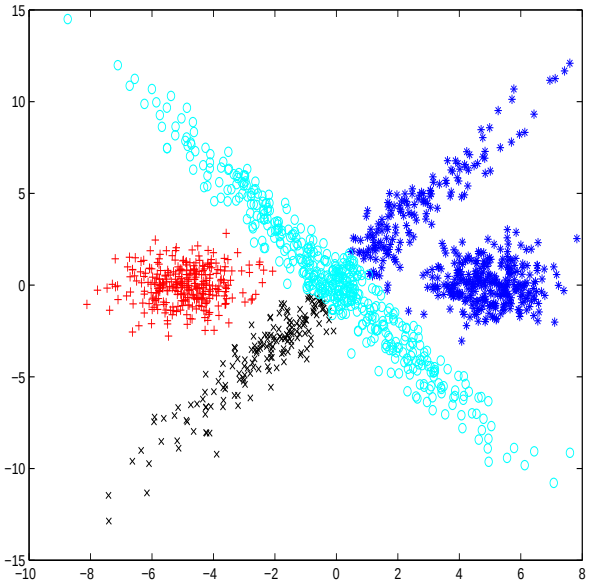


FIG. 3.9 – CEM : Sensibilité aux conditions d'initialisation (4 classes demandées)

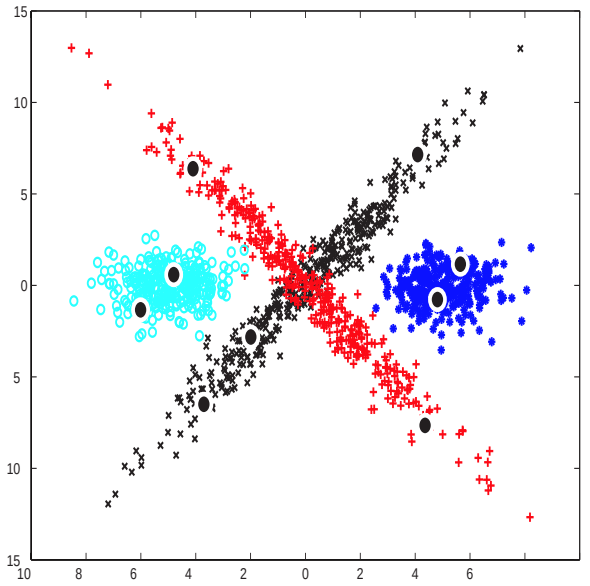


FIG. 3.10 – Correction par la supervision

la première fois l'emploi des méthodes de supervision partielle [Ped85]. Ce jeu de données comporte quatre classes où deux d'entre elles ont la même moyenne théorique et se croisent en leurs centres. Deux classes périphériques sont de chaque côté de la croix.

Après 5 initialisations aléatoires, la meilleure partition produite par l'algorithme CEM est représenté sur la figure (3.9). Toutefois, des initialisations répétées ont permis d'obtenir la partition correcte. L'idée de la semi-supervision est de contraindre le processus d'estimation en fixant les labels des points supervisés pour respecter la géométrie des classes (voir Fig. (3.10)).

Bien évidemment, superviser même partiellement permet d'améliorer la précision des différents estimateurs. Il en résulte une meilleure capacité en généralisation du modèle. Par suite, il a été montré que la semi-supervision permet de limiter le phénomène de Hughes dû à la dimensionalité des données [Hsi98].

3.2.2 Approche paramétrique de la semi-supervision

Ici, on fera la distinction terminologique entre les classes, catégories d'affectation du problème de décision, et les "clusters", groupements de points déterminés par un algorithme. L'expertise donne une information sur la classe et non sur le cluster des points supervisés. Les classes C_k seront indicées de 1 à c alors que les clusters $\bar{C}_l$ seront numérotés de 1 à $\bar{c}$. Le formalisme général présenté par la suite correspond à la recherche de classes multimodales : à plusieurs clusters peuvent correspondre une seule classe.

En notant y le numéro de la classe associée au point x de l'espace des caractéristiques, la densité conjointe $p(x, y)$ peut s'écrire comme le mélange suivant :

$$p(x, y) = \sum_{k=1}^{\bar{c}} P(\bar{C}_k)\ P(C_y|\bar{C}_k)\ f(x|\bar{C}_k) \tag{3.32}$$

$P(C_y|\bar{C}_k)$ est la probabilité a posteriori de la classe C_y au cluster $\bar{C}_k$ soumise à la contrainte $\sum_{y=1}^{c} P(C_y|\bar{C}_k) = 1$ pour tout $k = 1, \ldots, \bar{c}$. Celle-ci peut être interprétée en terme d'adéquation entre la classe C_y et le cluster $\bar{C}_k$. $f(x|\bar{C}_k)$ désigne la densité de probabilité en x relativement au cluster $\bar{C}_k$ (on supposera un modèle gaussien par la suite) ayant une probabilité a priori $P(\bar{C}_k)$ respectant la contrainte $\sum_{k=1}^{\bar{c}} P(\bar{C}_k) = 1$. Le vecteur Θ des paramètres du modèle s'écrit $\Theta \equiv [P(C_y|\bar{C}_k), \mu_k, \Sigma_k, P(\bar{C}_k) : y = 1, \ldots, c \text{ et } k = 1, \ldots, \bar{c}]^T$.

Dans ce cadre, la vraisemblance s'écrit grâce à la règle de Bayes :

$$\begin{aligned}
\mathcal{L}(\Theta) = P(\Theta|\chi) &= \frac{P(\Theta)P(\chi|\Theta)}{P(\chi)} \\
&= \overbrace{\frac{P(\Theta)}{P(\chi)}}^{\zeta} P(\chi|\Theta) \\
&= \zeta p(\chi^u|\Theta)\ p(\chi^d|\Theta) \\
&= \zeta \prod_{x_i \in \chi^u} p(x_i|\Theta) \prod_{(x_i,y_i)\in\chi^d} p(x_i, y_i|\Theta) \\
&= \zeta \prod_{x_i \in \chi^u} \sum_{k=1}^{\bar{c}} P(\bar{C}_k)\ f(x_i|\bar{C}_k) \times \\
&\quad \prod_{(x_i,y_i)\in\chi^d} \sum_{k=1}^{\bar{c}} P(\bar{C}_k)\ P(C_{y_i}|\bar{C}_k)\ f(x_i|\bar{C}_k)
\end{aligned} \tag{3.33}$$

Lorsqu'on souhaite maximiser la vraisemblance, on peut négliger le terme ζ, constant dans ce contexte. La log-vraisemblance est alors :

$$\begin{aligned}
\log \mathcal{L}(\Theta) &= \log(\zeta) + \sum_{x_i \in \chi^u} \log\left(\sum_{k=1}^{\bar{c}} P(\bar{C}_k)\ f(x_i|\bar{C}_k)\right) + \\
&\quad \sum_{(x_i,y_i)\in\chi^d} \log\left(\sum_{k=1}^{\bar{c}} P(\bar{C}_k)\ P(C_{y_i}|\bar{C}_k)\ f(x_i|\bar{C}_k)\right)
\end{aligned} \tag{3.34}$$

Les données manquantes z_{ik} correspondantes aux probabilités a posteriori des x_i dans les clusters $\bar{C}_k$ s'écriront :

- $P(\bar{C}_k|x_i) = P(\bar{C}_k)\ f(x_i|\bar{C}_k)\ /\ f(x)$ pour les données non étiquetées de χ^u ;
- $P(\bar{C}_k|x_i, y_i) = P(\bar{C}_k)\ P(C_{y_i}|\bar{C}_k)\ f(x_i|\bar{C}_k)\ /\ f(x)$ pour les données étiquetées de χ^d.

Après des calculs identiques à ceux effectués dans le contexte non supervisé, on voit que maximiser la vraisemblance complète équivaut à maximiser

$$\begin{aligned}
\log \mathcal{L}_c(\Theta) &= \sum_{x_i \in \chi^u} \sum_{k=1}^{\bar{c}} P(\bar{C}_k|x_i) \log(P(\bar{C}_k)\ f(x_i|\bar{C}_k)) + \\
&\quad \sum_{(x_i,y_i)\in\chi^d} \sum_{k=1}^{\bar{c}} P(\bar{C}_k|x_i, y_i) \log(P(\bar{C}_k)\ P(C_{y_i}|\bar{C}_k)\ f(x_i|\bar{C}_k))
\end{aligned} \tag{3.35}$$

L'algorithme 14 récapitule l'estimation les paramètres Θ du modèle.

Algorithme 14: L'algorithme EM semi-supervisé pour des classes gaussiennes multimodales

Entrée : $\chi^u = \{x_1, \ldots, x_{n_u}\}$ un ensemble de données n_u non étiquetées,
$\chi^d = \{(x_1, y_1), \ldots, (x_{n_d}, y_{n_d})\}$ un ensemble de n_d données étiquetées,
$\bar{c}$ le nombre de clusters recherchés, δ la fonction delta de Kronecker

Sortie : Une estimation $\hat{\Theta}$ des paramètres du modèle

Choisir les centres des clusters $\mu_k^{(0)}$ parmi les x_i ;
Initialiser les $\Sigma_k^{(0)}$ par la matrice de covariance estimée sur $\chi^u \bigcup \chi^d$;
Pour chaque classe C_y, calculer la probabilité a priori :

$$\hat{P}^{(0)}(C_y) = \frac{\sum_{(x_i,y_i)\in\chi^d} \delta_{y_i=y}}{n_d}$$

et pour chaque cluster k, la probabilité a posteriori $\hat{P}^{(0)}(C_y|\bar{C}_k) = \hat{P}^{(0)}(C_y)$;
répéter

E-Step (Estimation)
Calcul des probabilités a posteriori des données dans les clusters :

$$\hat{P}^{(t)}(\bar{C}_k|x_i) = \frac{\hat{P}^{(t)}(\bar{C}_k)\ f(x_i|\hat{\mu}_k^{(t)}, \hat{\Sigma}_k^{(t)})}{\sum_{k=1}^{\bar{c}} \hat{P}^{(t)}(\bar{C}_k)\ f(x_i|\hat{\mu}_k^{(t)}, \hat{\Sigma}_k^{(t)})} \tag{3.36}$$

$$\hat{P}^{(t)}(\bar{C}_k|x_i, y_i) = \frac{\hat{P}^{(t)}(\bar{C}_k)\ \hat{P}^{(t)}(C_{y_i}|\bar{C}_k)\ f(x_i|\hat{\mu}_k^{(t)}, \hat{\Sigma}_k^{(t)})}{\sum_{k=1}^{\bar{c}} \hat{P}^{(t)}(\bar{C}_k)\ f(x_i|\hat{\mu}_k^{(t)}, \hat{\Sigma}_k^{(t)})} \tag{3.37}$$

M-Step (Maximisation)
Probabilité a priori des clusters :

$$\hat{P}^{(t+1)}(\bar{C}_k) = \frac{\sum_{x_i\in\chi^u} \hat{P}^{(t)}(\bar{C}_k|x_i) + \sum_{(x_i,y_i)\in\chi^d} \hat{P}^{(t)}(\bar{C}_k|x_i, y_i)}{n_u + n_d} \tag{3.38}$$

Probabilité a posteriori des classes connaissant les clusters :

$$\hat{P}^{(t+1)}(C_y|\bar{C}_k) = \frac{\sum_{(x_i,y_i)\in\chi^d} \delta_{y_i=y} \hat{P}^{(t)}(\bar{C}_k|x_i, y_i)}{\sum_{(x_i,y_i)\in\chi^d} \hat{P}^{(t)}(\bar{C}_k|x_i, y_i)} \tag{3.39}$$

Moyennes et matrices de covariance des clusters :

$$\hat{\mu}_k^{(t+1)} = \frac{\sum_{x_i\in\chi^u} \hat{P}^{(t)}(\bar{C}_k|x_i)x_i + \sum_{(x_i,y_i)\in\chi^d} \hat{P}^{(t)}(\bar{C}_k|x_i, y_i)x_i}{\sum_{i=1}^{n_u} \hat{P}^{(t)}(\bar{C}_k|x_i) + \sum_{(x_i,y_i)\in\chi^d} \hat{P}^{(t)}(\bar{C}_k|x_i, y_i)} \tag{3.40}$$

$$\hat{\Sigma}_k^{(t+1)} = \frac{\sum_{x_i\in\chi^u} \hat{P}^{(t)}(\bar{C}_k|x_i)S_{ik} + \sum_{(x_i,y_i)\in\chi^d} \hat{P}^{(t)}(\bar{C}_k|x_i, y_i)S_{ik}}{\sum_{x_i\in\chi^u} \hat{P}^{(t)}(\bar{C}_k|x_i) + \sum_{(x_i,y_i)\in\chi^d} \hat{P}^{(t)}(\bar{C}_k|x_i, y_i)} \tag{3.41}$$
avec $S_{ik} = (x_i - \hat{\mu}_k^{(t+1)})(x_i - \hat{\mu}_k^{(t+1)})^T$

t ← t+1 ;

Le critère de convergence de l'algorithme peut être fondé sur la stabilité de la vraisemblance. Dans [LSH01], Larsen et al. proposent de stopper l'algorithme lorsque l'affectation des points aux clusters est stable entre deux itérations. Cette stratégie peut s'avérer suffisante dans le cas particulier d'un grand nombre de clusters et/ou de clusters bien séparés. Les expériences que nous avons menées ont montré que ce critère pouvait facilement être mis en défaut par la convergence lente de l'algorithme EM. Cela est particulièrement le cas lorsque les clusters sont peu séparés avec des proportions équivalentes. L'affectation finale des points aux clusters peut être effectuée selon la règle du MAP selon les probabilités a posteriori estimées $P(\bar{C}_k|x_i)$ et $P(\bar{C}_k|x_i, y_i)$.

Le principe de supervision partielle présenté dans cette partie est facilement applicable aux variantes d'EM. En supprimant la différenciation classes/clusters, les données étiquetées correspondent à la connaissance des probabilités a posteriori $P(\bar{C}_k|x_i)$. Pour l'algorithme CEM, il n'y a aucune modification à effectuer car l'étape de classification est déjà effectuée pour les points supervisés. Il suffit simplement de les inclure dans l'étape de maximisation. De la même façon, l'algorithme SEM peut être facilement adapté à la semi-supervision en n'effectuant pas l'étape de simulation pour les données étiquetées.

3.2.3 Approches géométriques de la semi-supervision

Dans cette section, les méthodes non probabilistes de la supervision partielle sont présentées.

3.2.3.1 Versions semi-supervisées des FCM

Historiquement, Pedrycz a introduit la notion de supervision partielle en modifiant l'algorithme des FCM [Ped85]. Le critère original (Eq. (2.22)) à minimiser est, rappelons-le :

$$J_f(U, V) = \sum_{k=1}^{c} \sum_{i=1}^{n} u_{ik}^m d_{ik}^2$$

$$\text{avec } \sum_{k=1}^{c} u_{ik} = 1 \text{ pour } i = 1, \ldots, n$$

où u_{ik} et d_{ik} désignent respectivement le degré d'appartenance et la distance de x_i à la classe C_k. Pedrycz propose de modifier cette fonctionnelle par l'introduction d'un terme de supervision (J_s) :

$$J_{ss} = \alpha_1 \underbrace{\sum_{k=1}^{c} \sum_{i=1}^{n} u_{ik}^2 d_{ik}^2}_{J} + \alpha_2 \underbrace{\sum_{k=1}^{c} \sum_{i=1}^{n} (u_{ik} - f_{ik} b_i)^2 d_{ik}^2}_{J_s} \tag{3.42}$$

Dans cette version, le coefficient de flou m est fixé à 2. Le terme J_s représente l'information liée à la semi-supervision. Le vecteur booléen b désigne les points supervisés ($b_i = 1$ si x_i est supervisé, 0 sinon). Dans le cas où le point x_i est supervisé, l'élément f_{ik} est le degré d'appartenance fixé par l'expert. La somme de degrés d'appartenance pour chacun de ces x_i doit respecter la contrainte de normalisation. Dans le cas contraire, on doit effectuer une normalisation. En revanche, il est impossible de prendre en compte l'information inverse : "Ce point n'appartient pas à cette classe." En effet, on peut considérer que l'information de supervision pour x_i n'est complète que si le vecteur f_i est entièrement donné par l'expert.

Dans l'équation (3.42), le terme $(u_{ik} - f_{ik}b_i)$ peut être interprété comme une mesure du désaccord entre le degré d'appartenance u_{ik} estimé par l'algorithme et le degré d'appartenance f_{ik} proposé par l'expert. La minimisation de la fonctionnelle J_{ss}, somme de terme positifs, va ainsi conduire à la minimisation de ce désaccord. Les coefficients α_1 et α_2 permettent d'effectuer un compromis entre l'influence des points supervisés et non supervisés. Pedrycz propose de fixer les valeurs de α_1 et α_2 par les formules :

$$\alpha_1 = \frac{1}{n_u}, \alpha_2 = \frac{1}{n_d}$$

où n_u et n_d désignent le nombre de points non supervisés et supervisés.

La minimisation sous contraintes de J_{ss} s'écrit en termes des multiplicateurs de Lagrange λ_i relatifs à la contrainte $\sum_{k=1}^{c} u_{ik} = 1$ pour chacun des points. On aboutit alors à la mise à jour suivante des degrés d'appartenance :

$$u_{ik} = \frac{2 - \sum_{l=1}^{c} f_{il} b_i}{2 \sum_{l=1}^{c} \frac{d_{ik}^2}{d_{il}^2}} + 0.5 f_{ik} b_i \tag{3.43}$$

Dans un article plus récent, Pedrycz et al. évoquent la forme générale du coefficient de flou m [PW97]. La fonctionnelle à minimiser devient :

$$J_{ss2} = \sum_{k=1}^{c} \sum_{i=1}^{n} u_{ik}^m d_{ik}^2 + \alpha \sum_{k=1}^{c} \sum_{i=1}^{n} (u_{ik} - f_{ik} b_i)^m d_{ik}^2 \tag{3.44}$$

Même si les auteurs se limitent ainsi au cas m=2, la formule générale pour les degrés d'appartenance s'écrit :

$$u_{ik} = \frac{1}{1 + \alpha^{1/(m-1)}} \left(\frac{1 + \alpha^{1/(m-1)} (1 - b_i \sum_{l=1}^{c} f_{il}) + \alpha^{1/(m-1)} f_{ik} b_i}{\sum_{l=1}^{c} \left(\frac{d_{ik}}{d_{il}} \right)^{2/(m-1)}} \right) \tag{3.45}$$

On remarquera que dans le cas non supervisé, la fonctionnelle J_{ss2} peut se réécrire sous la forme $J_{ss2} = (1 + \alpha) J_f$. Le terme $(1 + \alpha)$ est un facteur d'échelle qu'il convient d'ignorer

dans la minimisation. De même, la mise à jour des degrés d'appartenance dans l'équation (3.45) se simplifie pour donner la formulation classique des FCM :

$$u_{ik} = \left[\sum_{l=1}^{c} (\frac{d_{ik}}{d_{il}})^{\frac{2}{m-1}} \right]^{-1}$$

Algorithme 15: L'algorithme ssFCM (version Pedrycz)

Entrée : c le nombre de classes recherchées, ρ_k est le volume de la classe C_k, $d_{ik}^2 = ||x_i - v_k||_{A_k}^2$ la métrique employée, F matrice des degrés d'appartenance des points supervisés indiqués par le vecteur booléen b, α paramètre de contrôle de l'influence des points supervisés, m le coefficient de flou, ϵ le seuil de convergence de la partition

Sortie : Une partition floue U de χ

$t \longleftarrow 0$;

Initialiser U_0 en utilisant les degrés d'appartenance de F;

répéter

$t \leftarrow t + 1$;

{Etape 1 : Mise à jour des prototypes des classes}

pour *$k = 1$ à c* **faire**

$$v_{k,t} = \frac{\sum_{i=1}^{n} u_{ik,t-1}^m x_i}{\sum_{i=1}^{n} u_{ik,t-1}^m}$$

$$A_{k,t} = [\rho_k det(\Sigma_{k,t})]^{1/p} \Sigma_{k,t}^{-1} \text{ où } \Sigma_{k,t} = \frac{\sum_{i=1}^{n} u_{ik,t-1}^m (x_i - v_{k,t})(x_i - v_{k,t})^T}{\sum_{i=1}^{m} u_{ik,t-1}^m}$$

{Etape 2 : Mise à jour de la partition floue}

pour *$i = 1$ à n* **faire**

pour *$k = 1$ à c* **faire**

$$u_{ik,t} = \frac{1}{1 + \alpha^{1/(m-1)}} \left(\frac{1 + \alpha^{1/(m-1)}(1 - b_i \sum_{l=1}^{c} f_{il}) + \alpha^{1/(m-1)} f_{ik} b_i}{\sum_{l=1}^{c} \left(\frac{d_{ik,t}}{d_{il,t}} \right)^{2/(m-1)}} \right)$$

jusqu'à $||U_t - U_{t-1}|| < \epsilon$;

Concernant le choix de α, les auteurs proposent de choisir un terme proportionnel au rapport $\frac{n}{n_d}$. Par analogie avec l'algorithme RCA (version robuste compétitive de FCM), on peut imaginer faire varier ce seuil par une loi de décroissance. Dans les premières

itérations, on initialise les prototypes des classes en utilisant principalement les données supervisées (α fort). Cela peut être fait sous réserve quelles soient représentatives de la densité des classes. Au cours des itérations α décroît pour utiliser la structure des données non supervisées qui sont représentatives du mélange. L'algorithme 15 récapitule le fonctionnement général de la version semi-supervisée des FCM selon Pedrycz. La figure (3.11) montre la supériorité de l'algorithme ssFCM sur l'algorithme FCM sur la croix de Gustafson.

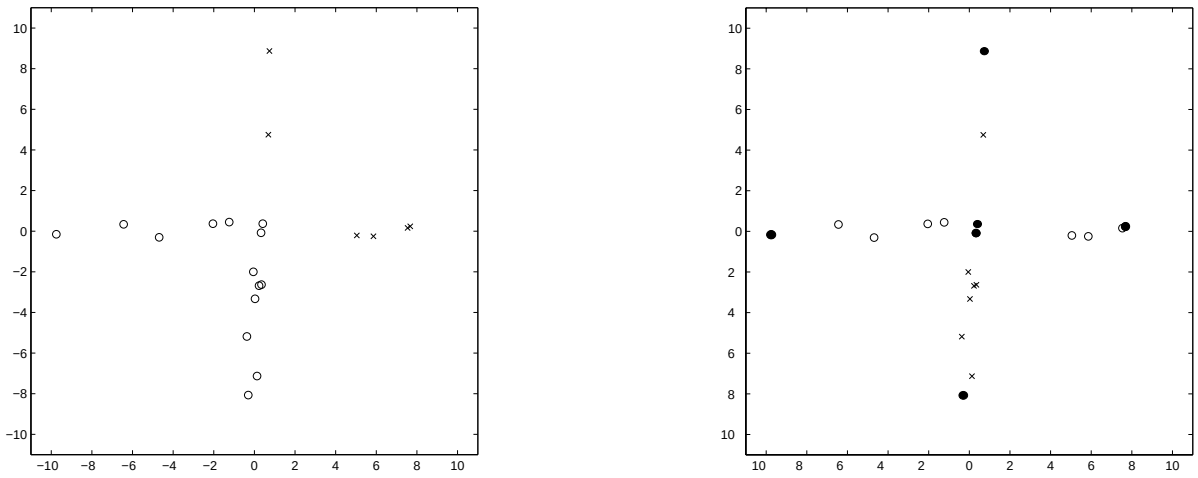


FIG. 3.11 – Croix de Gustafson : FCM échoue (à gauche) alors que ssFCM (version Pedrycz) retrouve la structure (à droite)

Le point important de la ssFCM de Pedrycz est que l'on recherche une partition U de l'ensemble des données, y compris sur des données supervisées. Si l'on choisit d'effectuer une affectation selon le maximum du degré d'appartenance pour les éléments de U, il est tout à fait possible de remettre en cause l'expertise des données.

Pour éviter ce problème, Bensaïd et al. proposent de restreindre la recherche de partition aux données non supervisées [BHBC96]. La matrice U peut être vue comme la jonction $[U^u|U^d]$ des matrices de partition U^u et U^d correspondant aux deux types de données. La matrice U^d, fixée tout au long de l'algorithme, contient l'information de supervision jouant le rôle de F pour l'algorithme précédent. On note u^d_{ik} le degré d'appartenance du i-ième point supervisé à la classe C_k. Par souci de simplification de la notation, l'indice concernant le numéro d'itération sera reporté en indice des variables. Ainsi, la valeur $u^u_{ik,t}$ désigne le degré d'appartenance de l'élément x_i non supervisé à la classe C_k lors de la t-ième itération de l'algorithme. Des conventions analogues sont prises pour les autres variables.

Le modèle de classe décrit par les auteurs correspond au cas de prototypes-point avec une métrique euclidienne pondérée : $d_{ik}^2 = ||x_i - v_k||_A^2$. Les prototypes des classes v_k sont initialisés à partir des données supervisées :

$$v_{k,0} = \frac{\sum_{i=1}^{n_d} (u_{ik}^d)^m x_i^d}{\sum_{i=1}^{n_d} (u_{ik}^d)^m} \tag{3.46}$$

Ce type d'initialisation traduit l'hypothèse sous-jacente de la représentativité des points supervisés vis à vis de la densité des classes. Par la suite, seule la matrice U^u des degrés d'appartenance des points non supervisés est mise à jour par la formule classique :

$$u_{ik,t}^u = \left[\sum_{l=1}^{c} \left(\frac{||x_i^u - v_{k,t-1}||_A}{||x_i^u - v_{l,t-1}||_A} \right)^{\frac{2}{m-1}} \right]^{-1} \tag{3.47}$$

A partir de ces valeurs, il est possible de recalculer les centres des classes. De manière à augmenter l'influence des points supervisés, un nouveau terme de pondération w_i est introduit, associé aux données supervisées. Bien évidemment, les w_i doivent être pris supérieurs à 1. L'approche de Bensaïd et al. consiste alors à modifier l'équation de mise à jour des prototypes par l'heuristique suivante

$$v_{k,t} = \frac{\sum_{i=1}^{n_d} w_i (u_{ik}^d)^m x_i^d + \sum_{i=1}^{n_u} (u_{ik,t}^u)^m x_i^u}{\sum_{i=1}^{n_d} w_i (u_{ik}^d)^m + \sum_{i=1}^{n_u} (u_{ik,t}^u)^m} \tag{3.48}$$

Cette équation n'est pas le résultat de la minimisation d'une fonctionnelle telle que dans le cas de FCM. Cette méthode de classification ne peut ainsi être considérée comme une véritable généralisation de FCM au cas semi-supervisé mais une méthode inspirée par FCM. Les auteurs ont constaté que l'estimation itérative converge d'autant plus vite que les poids w_i sont élevés. De plus, la méthode d'initialisation des prototypes, sujette au biais de sélection des points supervisés, nécessite de connaître au moins un point supervisé par classe. Cette hypothèse n'est pas toujours vérifiée en pratique. L'ensemble de la procédure est récapitulé par l'algorithme 16.

Les résultats obtenus par ces deux variantes semi-supervisées de FCM semblent donner des résultats équivalents sous réserve d'un paramétrage adéquat [BKKP99]. Les paramètres α et w_k influent de manière analogue dans le processus d'estimation en donnant plus ou moins d'importance aux données supervisées. Toutefois, les w_k sont des paramètres plus intuitifs que α et on peut contrôler ces pondérations pour chacun des points supervisés. On peut remarquer toutefois qu'il convient de ne pas les choisir trop grands car sinon, on biaise l'estimation des prototypes.

3.2.3.2 Approche hiérarchique de la semi-supervision

La semi-supervision peut être également utilisée dans le cadre de la classification hiérarchique. Bensaïd et al. ont proposé une méthode fondée sur le sur-partitionnement se

Algorithme 16: L'algorithme ssFCM (version Bensaïd)

Entrée : c le nombre de classes recherchées, U^d matrice de partition des données supervisées fixée par l'expert, U_0^u matrice de partition des données supervisées respectant la contrainte $\sum_{k=1}^{c} u_{ik,0}^u = 1$ pour $i = 1, \ldots, n_u$, $w = \{w_1, \ldots, w_{n_d}\}$ ensemble des pondérations de points supervisés, ϵ seuil de convergence de la matrice de partition U^u

Sortie : Une partition floue $U = [U^u|U^d]$ de χ^u

$t \longleftarrow 0$;

{Calcul des prototypes initiaux à partir des données supervisées}

pour *$k = 1$ à c* **faire**

$$v_{k,0} = \frac{\sum_{i=1}^{n_d} (u_{ik}^d)^m x_i^d}{\sum_{i=1}^{n_d} (u_{ik}^d)^m}$$

répéter

$t \leftarrow t + 1$;

{Mise à jour de la matrice de partition U^u}

pour *$i = 1$ à n_u* **faire**

pour *$k = 1$ à c* **faire**

$$u_{ik,t}^u = \left[\sum_{l=1}^{c} \left(\frac{||x_i^u - v_{k,t-1}||_A}{||x_i^u - v_{l,t-1}||_A} \right)^{\frac{2}{m-1}} \right]^{-1}$$

{Calcul des prototypes}

pour *$k = 1$ à c* **faire**

$$v_{k,t} = \frac{\sum_{i=1}^{n_d} w_i (u_{ik}^d)^m x_i^d + \sum_{i=1}^{n_u} (u_{ik,t}^u)^m x_i^u}{\sum_{i=1}^{n_d} w_i (u_{ik}^d)^m + \sum_{i=1}^{n_u} (u_{ik,t}^u)^m}$$

jusqu'à $||U_t^u - U_{t-1}^u|| < \epsilon$;

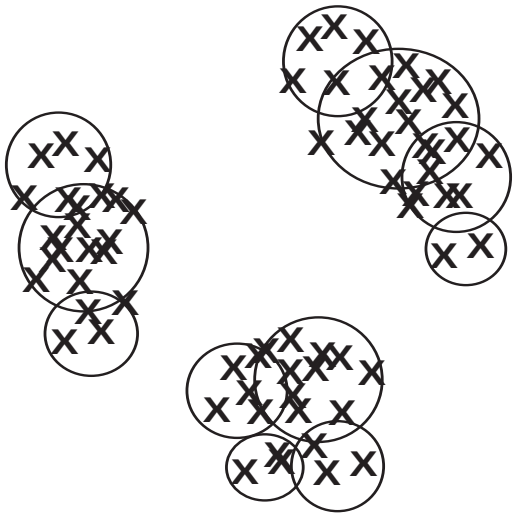


FIG. 3.12 – Sur-partitionnement en n_d classes

déroulant en 4 étapes [BB98][ALB97] et produisant une partition possibiliste. L'idée sous-jacente à cette méthode est d'associer la même classe à un ensemble de points que celle du point supervisé le plus proche. Pour faire une analogie avec l'algorithmique géométrique, on retrouve grossièrement le principe du diagramme de Voronoï.

La première phase de la méthode consiste à sur-partitionner l'ensemble des points en autant de clusters qu'il existe de points supervisés (voir Fig. (3.12)). Ici, l'étiquette des points supervisés n'est pas prise en compte mais on profite de leur disponibilité pour améliorer l'estimation de la densité mélange. Ce partitionnement peut être effectué avec n'importe quel algorithme de classification automatique. Cependant, il est préférable d'utiliser une méthode simple dont la complexité dépend peu ou pas du nombre de classes recherchées. En effet, si l'on dispose d'un grand nombre de points supervisés par rapport au nombre total de points, le coût des méthodes itératives telles que FCM ou EM peut être prohibitif. Ainsi, Amar et al. proposent d'utiliser une méthode de classification hiérarchique dans laquelle on coupe la hiérarchie de manière à obtenir n_d classes [ALB97]. On notera $V = \{v_1, \ldots, v_{n_d}\}$ l'ensemble des prototypes-point déterminés par l'algorithme de classification automatique. Dans le cas d'une classification hiérarchique, les prototypes-point peuvent être définis à partir des centres de gravité des clusters. Par la suite, on désigne par U^* la matrice de partition de taille $n_d \times n_u$ décrivant l'appartenance des points non supervisés à chacun des clusters. Les vecteurs d'appartenance des points supervisés seront fixés par l'expert dans la matrice $F = [f_1^T, \ldots, f_{n^d}^T]^T$ de terme général f_{ik}. Une fois le partitionnement effectué, on va chercher une mesure de similarité entre les classes réelles et les clusters issus du partitionnement. La matrice L de dimension $c \times n_d$ désignera pour un élément $l_{k,l}$ l'adéquation entre la classe C_k et le cluster SC_l.

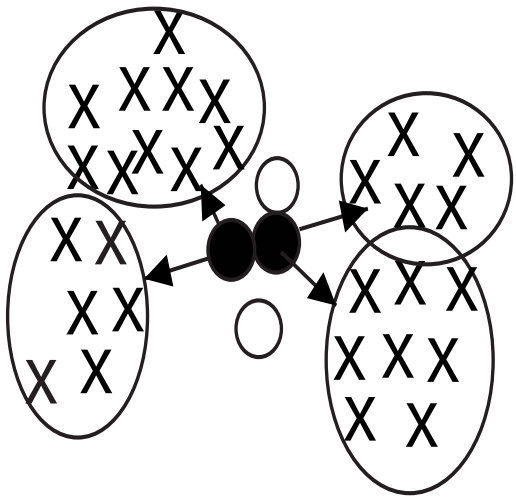


FIG. 3.13 – Problème de la recopie des degrés d'appartenance pour ssPPC - seule la classe des points noirs est représentée dans la matrice L

Dans l'algorithme originel [BB98] et [ALB97], Bensaïd et al. proposent de réduire ce problème à une distance entre les points supervisés et les sous-classes. Il s'agit pour un cluster donné de recopier les degrés d'appartenance du point supervisé le plus proche :

$$L = [f^T_{nn(1)}, f^T_{nn(2)}, \ldots, f^T_{nn(n_d)}]$$

où $nn(l) = arg\min_{i\in 1,\ldots,n_d} d(x^d_i, SC_l)$. La distance $d(x^d_i, SC_l)$ est choisie parmi les différents types de lien utilisés en classification hiérarchique : lien minimum, lien complet, lien moyen (voir section 2.3.2). Cette approche peut être mise en défaut par une configuration particulière des points supervisés qui conduit à produire une partition dégénérée[10][LBB98] (voir Fig. 3.13). Tazi et al. proposent ainsi d'utiliser deux nouvelles mesures [LBB98] garantissant une partition finale non dégénérée :

$$l_{kl} = \frac{1 - \frac{d(C_k,SC_l)}{\sum_{m=1}^{c} d(C_k,SC_m)}}{c-1} \quad (3.49)$$

$$\text{avec } d(C_k, SC_l) = \min_{x^d_i \in C_k} ||x^d_i - v_l|| \quad (3.50)$$

et

$$l_{kl} = \frac{|S_l|}{|\chi^d_k|} \quad (3.51)$$

$$\text{où } S_l = \{x^d_i \in \chi^d_k \text{ tel que } ||x^d_i - v_l|| \leq ||x^d_i - v_m||, \forall m \neq l\} \quad (3.52)$$

On peut remarquer que dans chacune des trois variantes pour le calcul de L, il est nécessaire de connaître au moins un point supervisé par classe. Cette hypothèse n'est pas

[10]Il existe au moins une classe ne contenant aucun point à la fin de l'algorithme

toujours vérifiée en pratique lorsque l'expertise est coûteuse.

La dernière étape consiste à calculer la partition finale des points non supervisés en effectuant le produit matriciel $U^u = LU^*$. Ainsi, les clusters déterminés dans l'étape 1 sont pondérés par leur adéquation aux classes réelles représentées par les données supervisées. Pour respecter les contraintes du cadre possibiliste, les degrés d'appartenance sont majorés à 1. On écrit ainsi :

$$U^u_{ik} = min(1, < L_i, U^*_k >) \tag{3.53}$$

où $< ., . >$ désigne l'opérateur produit scalaire. L'algorithme 17 reprend les diverses étapes de la méthode.

Algorithme 17: Un algorithme semi-supervisé de classification avec sur-partitionnement

Entrée : $\chi^u = \{x_1, \ldots, x_{n_u}\}$ un ensemble de données n_u non étiquetées,
$\chi^d = \{x^d_1, \ldots, x^d_{n_d}\}$ un ensemble de n_d données étiquetées avec au moins un point supervisé par classe, c le nombre de classes

Sortie : Une partition possibiliste U^u de χ^u

Etape 1 : Le sur-partitionnement

Sur-partitionner $\chi^u \bigcup \chi^d$ en n_d clusters SC_k, $(k = 1, \ldots, n_d)$ en utilisant un algorithme de classification automatique.

Stocker la partition résultante dans U^* et les prototypes dans $V = \{v_1, \ldots, v_{n_d}\}$

Etape 2 : L'adéquation entre les classes réelles et les clusters

Déterminer la matrice L où L_{kl} désigne l'adéquation entre la classe C_k et le cluster SC_l en utilisent les équations (3.49) ou (3.51).

Etape 3 : Calculer la matrice de partition possibiliste finale

$$U^u_{ik} = min(1, < L_i, U^*_k >)$$

Les auteurs ont testé cet algorithme sur le jeu de données Iris [BM98a]. Les résultats obtenus laissent supposer une supériorité de la méthode des plus proches prototypes (Eq. 3.51).

L'approche paramétrique présentée dans la section 3.2.2 peut être utilisée conjointement avec une démarche hiérarchique. Récemment, Larsen et al. ont proposé un algorithme de classification semi-supervisée hiérarchique fondé sur des modèles de mélanges [LSH01]. Cet algorithme peut se décomposer en trois étapes :

– **Étape 1** : Sur-partitionner les données en $\bar{c}$ clusters. $\bar{c}$ doit être supérieur au nombre final de classes. (voir section 3.2.2).

- **Étape 2** : Effectuer une classification hiérarchique agglomérative à partir K clusters jusqu'a en obtenir un seul.
- **Étape 3** : Couper la hiérarchie pour obtenir la partition finale selon le nombre de classes optimal pour un critère ou fixé par l'utilisateur.

Une fois l'étape 1 effectuée, on dispose d'une estimation $\hat{\Theta}$ du vecteur de paramètres $\Theta \equiv [P(y|k), \mu_k, \Sigma_k, P(k) : \forall y, k]$. Chaque cluster C_k est modélisé par le triplet $\Theta_k = (P(k), \mu_k, \Sigma_k)$.
Dans l'étape 2, la fusion entre deux clusters C_k et C_l est décidée à l'aide d'une distance entre Θ_k et Θ_l. Plusieurs choix sont possibles :

- Minimiser la distance L_2 :

$$D(C_k, C_l) = \int (P(C_k)p(X|C_k) - P(C_l)p(X|C_l))^2 dx \tag{3.54}$$

- Maximiser la confusion entre les clusters

$$Erreur(C_k, C_l) = \int_{R_l} P(C_k)p(X|C_k)dx + \int_{R_k} P(C_l)p(X|C_l)dx \tag{3.55}$$

  où R_k et R_l sont respectivement les zones d'affectation de x à C_k et C_l selon le critère MAP.
- La divergence de Kullback-Liebler (version symétrique) :

$$D(C_k, C_l) = \int p(x|C_k) \log(\frac{p(x|C_l)}{p(x|C_k)})dx + \int p(x|C_l) \log(\frac{p(x|C_k)}{p(x|C_l)})dx \tag{3.56}$$

Cette dernière distance est difficile à mettre en pratique lorsque les clusters à évaluer sont multimodaux. De plus, il est nécessaire de prendre en compte la probabilité a priori des clusters $P(C_k)$ afin de ne pas pénaliser les clusters denses (variance faible) (voir [LSH01]).

3.2.4 Autres approches de la semi-supervision

3.2.4.1 Machines à vecteurs de support semi-supervisées

Les Machines à Vecteurs de Support (MVS) sont principalement issues des travaux de Vapnik à la fin des années 70 [Vap82][Vap95]. Restés confidentiels pendant une dizaine d'années, ces travaux suscitent actuellement un grand intérêt du fait des bons résultats obtenus par les MVS. [OFG97][Joa98][PV98]. Contrairement aux méthodes traditionnelles qui cherchent à minimiser l'erreur sur l'ensemble d'apprentissage, Vapnik propose de la remplacer par la minimisation du risque structurel (voir [Bur98]). Ce principe repose grossièrement sur la minimisation conjointe du risque empirique et de la complexité du modèle apprenant. Si l'on souhaite seulement minimiser l'erreur de classification sur l'ensemble d'apprentissage, on peut utiliser un modèle complexe (Ex. : polynômes de degré élevé). Dans ce cas là, les frontières de décision pourront être apprises par coeur (voir Fig.

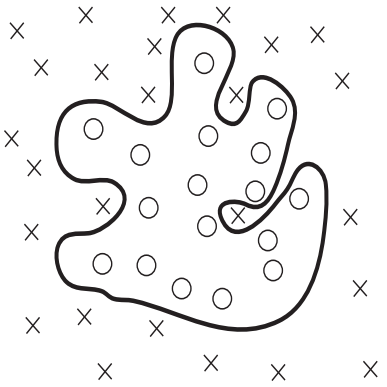


FIG. 3.14 – Modèle trop libre

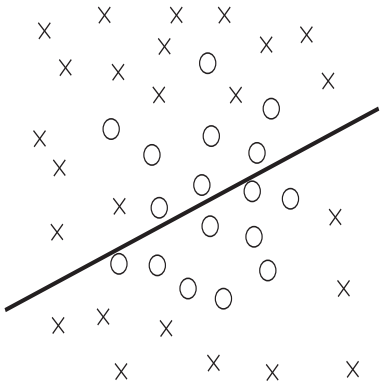


FIG. 3.15 – Modèle trop contraint

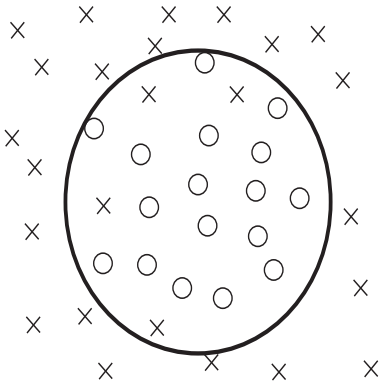


FIG. 3.16 – Modèle optimal

3.14). Malheureusement, la capacité en généralisation du discriminateur en est réduite d'autant. A l'inverse, si l'on choisit un modèle trop simple (Ex. : séparation linéaire), l'erreur empirique est peut être grande (voir Fig. 3.15). Ces problèmes ne sont pas exclusifs aux machines à vecteurs de support puisqu'on les retrouve avec d'autres discriminateurs tels que les réseaux de neurones ou les k plus proches voisins (pour k faible). Il est donc nécessaire de faire un compromis entre erreur d'apprentissage et complexité du modèle employé.

Les machines à vecteurs de support ont été introduites dans le contexte de la classification supervisée. La présentation qu'il en est faite ici se restreindra à la recherche de frontières linéaires dans un problème à deux classes. La comparaison de différentes approches multi-classes est abordé dans [HL01]. Par commodité de notation, les classes sont généralement numérotées -1 , 1. On dispose ainsi d'un ensemble de couple points-classes $\chi = \{(x_1, y_1), \ldots, (x_{n_d}, y_{n_d})\}$. Dans le cas où les classes sont séparables par un hyperplan, l'équation de celui-ci peut s'écrire $w.x + b = 0$ où w est un vecteur w et b un scalaire. La contrainte de séparabilité sur les (x_i, y_i) peut être résumée par

$$y_i(w.x_i - b) \geq 1, \text{pour tout } i = 1, \ldots, n \tag{3.57}$$

Si l'on note d^+(resp. d^-), la distance minimale d'un point de la classe "1" (resp. "-1") à l'hyperplan séparateur, le principe de maximisation du risque structurel revient à de maximiser la marge $d = d^+ + d^- = \frac{2}{||w||}$ séparatrice des deux classes afin d'obtenir la meilleure généralisation possible. $||w||$ désigne la norme euclidienne du vecteur w, normal aux hyperplans définis précédemment (voir figure (3.17)). Cette maximisation peut se transformer en un problème de minimisation de $\frac{||w||^2}{2}$ sous les contraintes définies par

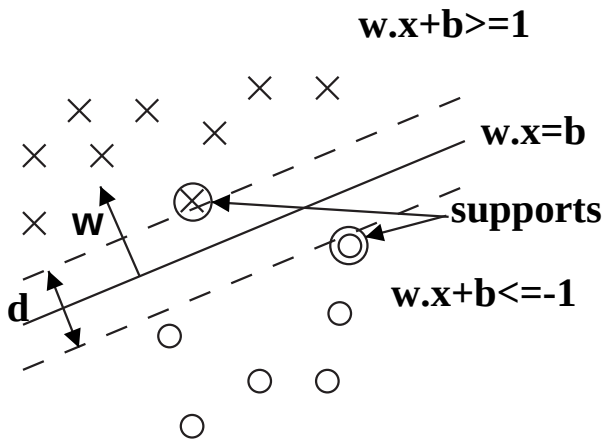


FIG. 3.17 – Maximisation de la marge dans les machines à vecteurs de support

l'équation (3.57) dont la formulation lagrangienne est

$$L(w,b,\lambda) = \frac{1}{2}w.w - \sum_{i=1}^{n} \lambda_i (y_i(w.x_i - b) - 1) \tag{3.58}$$

où les λ_i sont les multiplicateurs de Lagrange associés à chacun des points x_i ($\lambda = [\lambda_1, \ldots, \lambda_n]$). En annulant les dérivées partielles par rapport à w et b, on en déduit que la solution w^* s'exprime comme une combinaison linéaire des x_i : $w^* = \sum_{i=1}^{n} \lambda_i y_i x_i$. Ainsi, les points x_i tels que les λ_i sont strictement supérieurs à 0 seront appelées vecteurs de support. Ces vecteurs de support appartiennent aux deux hyperplans $y_i(w.x_i - b) = 1$. La terminologie "vecteur de support" est particulièrement bien appropriée puisqu'il suffit d'en supprimer un pour modifier la frontière de décision. La résolution effective du problème d'optimisation sous contraintes peut être effectuée avec un solveur de type CPLEX[11][CPL94].

Dans le cas où les données ne sont pas linéairement séparables, l'équation précédente n'a pas de solution. L'idée est de relâcher les contraintes sur les données en introduisant un terme d'erreur $\xi_i \geq 0$. Les contraintes de l'équation (3.57) deviennent :

$$y_i(w.x_i - b) + \xi_i \geq 1 \text{ pour tout } i = 1, \ldots, n \tag{3.59}$$

$$\xi_i \geq 0 \text{ pour tout } i = 1, \ldots, n \tag{3.60}$$

On peut remarquer qu'une erreur de classification n'interviendra que si $\xi_i > 1$. Ainsi, on peut évaluer une borne supérieure de l'erreur par $\sum_{i=1}^{n} \xi_i$. Comme on souhaite également

[11] Dans [BD99], Bennett et al. reformulent ce problème d'optimisation en terme de programmation linéaire. Ceci permet l'utilisation d'algorithmes de résolution plus efficaces.

minimiser cette valeur, la fonctionnelle précédente devient :

$$L(w,b,\xi,\lambda,\beta) = \frac{1}{2}w.w + \alpha\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i(y_i(w.x_i - b) + \xi_i - 1) - \sum_{i=1}^{n}\beta_i\xi_i \qquad (3.61)$$

Le terme α est un terme de pénalité qui permet de contrôler la capacité en généralisation par l'influence donnée aux erreurs. Les β_i sont les multiplicateurs de Lagrange associés aux contraintes $\xi_i \geq 0$. Après quelques calculs, on obtient des solutions de la même forme que dans le cas séparable mais dépendant du terme α.

Bien que les machines à vecteurs soient initialement fondées sur un apprentissage totalement supervisé, plusieurs versions semi-supervisées ont été proposées ces dernières années [BD99] [FM00]. Nous n'en présenterons ici qu'une seule version.
Une idée intuitive de la semi-supervision pour les MVS est de construire la meilleure machine pour les données supervisées tout en minimisant le nombre de données non étiquetées dans la marge (entre les deux hyperplans). Pour un point x_i, être dans la marge implique $|w.x_i - b| < 1$. Soient τ_i et υ_i les erreurs de classement commises sur les points non supervisés :

- classement à tort dans "+" : $(w.x_i - b) + \tau_i \geq 1$
- classement à tort dans "-" : $-(w.x_i - b) + \upsilon_i \geq 1$

Le critère à minimiser peut s'écrire alors comme dans [BD99] :

$$\min_{w,b,\xi,\tau,\upsilon} ||w|| + \alpha\left[\sum_{i=1}^{n_d}\xi_i + \sum_{j=n_d+1}^{n_d+n_u} min(\tau_j,\upsilon_j)\right] \qquad (3.62)$$

Avec les contraintes,

$$\begin{cases} y_i(w.x_i + b) + \xi_i \geq 1 & \xi_i \geq 0 \quad i = 1,\ldots,n_d \\ w.x_j - b + \tau_j \geq 1 & \tau_j \geq 0 \quad j = n_d+1,\ldots,n_d+n_u \\ -(w.x_j - b) + \upsilon_j \geq 1 & \upsilon_j \geq 0 \end{cases}$$

La principale difficulté de mise en oeuvre des machines à vecteurs de support se situe dans le temps de calcul nécessité par l'optimisation quadratique sous un grand nombre de contraintes. A ce jour, la taille des problèmes dépasse rarement les 10^5 points. De plus, il n'est pas sûr que l'introduction de données non supervisées et les contraintes qui leur sont associées permettent d'accélérer la convergence de la méthode d'optimisation.

3.2.4.2 Utilisation d'un critère d'information

Comme il a été vu précédemment, la semi-supervision peut être effectuée en contraignant un algorithme de classification non supervisée. Dans le cas des FCM, il s'agissait de forcer les degrés d'appartenance des points supervisés et par la même ceux des points non supervisés (ssFCM - approche Pedrycz). Une démarche similaire peut être entreprise en

considérant des critères d'information. L'idée est de pénaliser la fonctionnelle d'un algorithme de classification automatique par une mesure d'adéquation des données étiquetées aux classes produites :

$$\beta \text{ Critère-classification}(\chi^u, \chi^d) + \alpha \text{ Adéquation}(\chi^d)$$

où β et α sont des coefficients qui pondèrent l'influence relative des deux termes. L'une des mesures d'adéquation possibles est l'indice de Gini communément utilisé dans les méthodes de classification par arbres de décision :

$$G = \sum_{k=1}^{c} \sum_{l=1, l\neq k}^{c} P(C_k)P(C_k|C_l) \tag{3.63}$$

Cet indice, qui est une moyenne pondérée de l'erreur de classement, sera d'autant plus élevé que les classes sont impures. Les valeurs $P(C_k|C_l)$ et $P(C_k)$ sont généralement estimées par une approche fréquentiste : $\hat{P}(C_k|C_l) = \frac{Conf_{k,l}}{N_k}$ $\hat{P}(C_k) = \frac{N_k}{N}$ où $N_k = Card(C_k)$ et $Conf_{k,l}$ désigne le nombre de points χ_d appartenant à la classe l classés dans C_k. On remarquera que l'utilisation de ce critère nécessite une partition stricte des données.

Demiriz et al. ont proposé d'utiliser ce principe en effectuant l'optimisation par un algorithme génétique [DBE99]. L'algorithme de classification utilisé étant les C-moyennes, la fonctionnelle à minimiser s'écrit :

$$\beta \sum_{k=1}^{c} \sum_{x_i \in C_k} ||x_i - \mu_k||^2 + \alpha \sum_{k=1}^{c} \sum_{l=1, l\neq k}^{c} \frac{Conf_{k,l}}{N} \tag{3.64}$$

Les algorithmes génétiques sont particulièrement adaptés à l'optimisation de fonctions non linéaires avec des nombreux minima locaux [Hol75] [Gol89]. De nombreux autres critères d'adéquation sont disponibles sachant qu'on cherche à comparer deux distributions : celle estimée à partir des points supervisés et celle provenant de l'algorithme de classification automatique. Ainsi, on peut utiliser :

- le gain d'information (utilisé dans C4.5) [Qui93]
- la distance de Kullback-Liebler [Kul59]
- le test du Chi-deux

Conclusion

Dans ce chapitre, nous avons présenté deux aspects de la classification : la robustesse, la semi-supervision. L'étude de méthodes robustes de classification a montré qu'il existait trois manières distinctes de traiter les points aberrants :

- Utilisation d'estimateurs robustes tout en gardant une modélisation classique de classes (Ex. : RCA)

- Modélisation explicite de l'appartenance à une classe de bruit supplémentaire (Ex. : FNC)
- Utilisation d'un modèle de contamination pour la modélisation théorique des classes (Ex. : mélange de t-distribution)

Ces diverses approches donnent des résultats comparables et il convient de choisir en fonction du rapport de la difficulté du paramétrage à la complexité. Concernant cet aspect, l'algorithme EM pour un mélange de t-distribution est intéressant car il permet d'ajuster automatiquement le modèle aux points par la mise à jour des degrés de liberté. En revanche, cette méthode converge très lentement.

Concernant la semi-supervision, nous avons fait un état de l'art des méthodes actuelles. Nous avons ainsi vu que superviser partiellement revient généralement à contraindre l'estimation des paramètres des classes par une pondération plus forte des points supervisés.

Chapitre 4

Une méthode de classification robuste semi-supervisée

Dans ce chapitre, nous proposons un algorithme de classification robuste semi-supervisée avec rejet. Dans un premier temps, nous présentons le principe d'une estimation robuste itérative, son application au calcul des paramètres des classes en présence de données aberrantes. La deuxième partie est consacrée au développement d'un algorithme de classification robuste fondé sur un modèle de contamination des données. A partir de ce modèle, nous définissons une règle d'affectation avec rejet des points aberrants. Enfin, une modification de l'algorithme est apportée dans le but d'intégrer les concepts de supervision partielle. Tout au long du chapitre, des expérimentations sont menées, chacune étant focalisée sur un aspect particulier. Les aspects traités sont abordés de manière progressive.

4.1 Estimation robuste itérative

4.1.1 Principe

L'estimation robuste présentée dans le chapitre précédent peut être utilisée dans le cas d'une estimation robuste itérative [HW77] [RL87]. Cette méthode est fondée sur une repondération des erreurs à chaque itération de l'estimation ("Reweigthed Least Squares"). Dans ce cadre, la formulation du problème de régression peut alors s'écrire :

$$E_{\rho}^{(\tau)}(\theta) = \sum_{i=1}^{n} w_i^{(\tau-1)} e_i^{(\tau)}(\theta) \tag{4.1}$$

où $w_i^{(\tau-1)} = \rho(e_i^{(\tau-1)}(\theta))$ désigne un terme de pondération de l'erreur courante $e_i^{(\tau)}$. Le cas particulier $w_i = 1$ correspond à une régression au sens des moindres carrés. La procédure d'estimation est initialisée par un moindre carré standard. L'hypothèse sous-jacente est

qu'une donnée aberrante x_i possède une erreur forte car mal modélisée. Comme on souhaite réduire l'influence de ces données, il convient de choisir une fonction ρ décroissante en fonction de l'erreur. Ce choix peut être effectué parmi les M-estimateurs de la table (3.1) (Médiane, Cauchy, Huber, Tuckey). On réitère le processus jusqu'à convergence de la suite $\{\theta^{(\tau)}\}$. Cependant, il est généralement nécessaire d'effectuer un compromis entre la robustesse, la précision et la complexité de la méthode. En effet, au cours des itérations, le nombre de points pris en compte dans l'estimation peut décroître jusqu'à 1. La proportion des points dont le poids est proche de zéro est alors forte. De plus, la complexité de la méthode passe de $\Theta(n)$ à $\Theta(\tau n)$ [12]. Il convient donc de définir un critère d'arrêt qui peut être une combinaison de plusieurs types :

1. $\tau < \tau_{max}$ où τ désigne le nombre d'itérations effectuées,
2. convergence de la suite $\{\theta^{(\tau)}\}$,
3. taux maximal d'élimination $\alpha < \alpha_{max}$ où α est le pourcentage des données ayant un poids quasi-nul.

Concernant le choix de α_{max}, on doit garantir que l'on dispose d'un nombre minimum des points non éliminés pour estimer les paramètres du modèle (Ex. : $\frac{p(p+1)}{2}$ pour une matrice de covariance)). Nous allons illustrer la méthode d'estimation décrite par le calcul d'une moyenne et matrice de covariance robustes.

4.1.2 Application au calcul d'une moyenne et d'une matrice de covariance

Le principe d'estimation itérative présentée précédemment est applicable dans le cas de l'estimation robuste d'une moyenne (voir algorithme 18).

Afin de tester cette procédure, nous avons produit un ensemble χ de 22 nombres réels dont 20 sont distribués selon $\mathcal{N}(20, 2)$ et 2 égaux à 100 et 500. Les nombres 100 et 500 peuvent être considérés comme des points aberrants. La moyenne empirique est $\bar{\chi} = 45.06$ alors que la médiane vaut $med(\chi) = 19.4986$. Bien évidemment, on constate que la moyenne théorique est plus perturbée que la médiane, confirmant par la même la différence entre les points de rupture des 2 estimateurs (voir théorèmes (1) et (2)). Nous allons maintenant utiliser la procédure de calcul itératif robuste avec le M-estimateur de Huber. L'objectif de cette expérimentation est de démontrer que le calcul de la moyenne par cette procédure est au moins aussi précis que la moyenne expérimentale si l'on fait varier le seuil du M-estimateur. La question du choix de ce seuil sera abordée par la suite. La procédure itérative est stoppée lorsque la moyenne estimée est identique pour deux itérations successives (critère de convergence). La figure (4.1) représente la valeur moyenne estimée en fonction du seuil h.

[12] Il s'agit de la complexité en moyenne car il n'existe pas de pire des cas

Algorithme 18: Calcul itératif robuste de la moyenne à l'aide d'un M-estimateur

Données : $\chi = \{x_1, \ldots, x_n\}$, ψ la fonction d'influence du M-estimateur
Sortie : $\tilde{\mu}$, une estimation robuste de la moyenne
$\tau \leftarrow 0$;
$\tilde{\mu}^{(0)} = \bar{\chi}$;
répéter
  pour *i = 1 à n* **faire**
    $e_i^{(\tau-1)} = x_i - \tilde{\mu}^{(\tau-1)}$;
    $w_i^{(\tau-1)} = \frac{\psi(e_i^{(\tau-1)})}{e_i^{(\tau-1)}}$
  $\tilde{\mu}^{(\tau)} = \frac{\sum_{i=1}^{n} w_i^{(\tau-1)} x_i}{\sum_{i=1}^{n} w_i^{(\tau-1)}}$;
  $\tau \leftarrow \tau + 1$;
jusqu'à *Critère d'arrêt ou de convergence*;

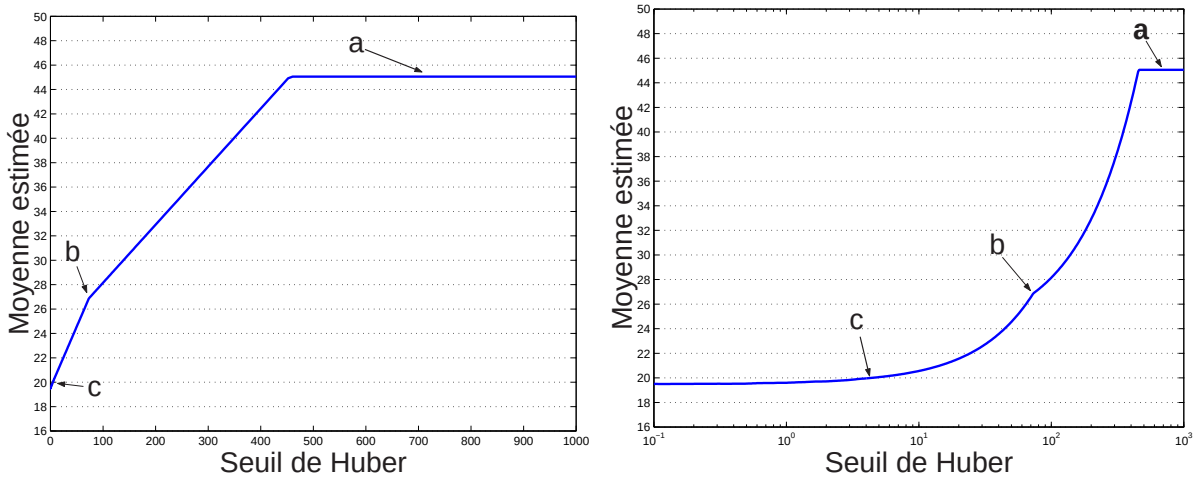


FIG. 4.1 – Moyenne estimée en fonction du seuil - Échelle linéaire à gauche, logarithmique à droite

On peut faire plusieurs remarques :

- (a) Lorsque h tend vers l'infini, la moyenne estimée devient la moyenne arithmétique. En fait, cela revient à accorder un poids maximal de 1 à tous les points. Si l'on note $\tilde{\chi}$ la moyenne robuste estimée sur χ et dépendante de h, on a :

$$\lim_{h \to \infty} \tilde{\chi} = \bar{\chi}$$

- (b) La valeur estimée croît quasi linéairement en fonction du seuil de Huber car la fonction de pondération est de la forme $\frac{h}{|e_i|}$. Le poids d'un point aberrant augmente en même temps que h. Pour la même raison, les points d'infléchissement de la courbe ne sont pas des ruptures nettes. Les échelles relativement différentes des deux axes

laissent supposer une forte sensibilité au choix de h. Cependant la pente réelle entre (b) et (c) est environ 0.1.
- (c) La valeur optimale du seuil de Huber est 4.39 (un peu plus de 2 fois l'écart-type). Pour une valeur plus petite de ce seuil, l'estimation prend en compte trop peu de points.
- L'algorithme converge en peu d'itérations (ici 4 au maximum).

Ce type d'estimation peut être adapté à l'estimation simultanée de la moyenne et de la matrice de covariance. Pour cela, on procède en trois étapes :
- Estimation de l'erreur e_i en termes de la distance de Mahalanobis au centre de la classe :
$$d^2_{\mathcal{M}}(x;\mu,\Sigma) = (x_i-\mu)^T\Sigma^{-1}(x_i-\mu); \tag{4.2}$$
- Calcul des poids w_i en fonction de l'erreur :
$$w_i = \frac{\Psi(e_i)}{e_i}$$
où Ψ est la fonction d'influence d'un M-estimateur (derivée première de l'estimateur)
- Ré-estimation la moyenne et matrice de covariance pondérée.

Cette procédure est détaillée dans l'algorithme 19.

Algorithme 19: Calcul itératif robuste de la moyenne et de la matrice de covariance

Données : $\chi = \{x_1,\dots,x_n\}$, ψ la fonction d'influence du M-estimateur

Sortie : $\tilde{\mu}$, $\tilde{\Sigma}$ une estimation robuste de la moyenne et de la matrice de covariance

$\tau \leftarrow 0$;

$\tilde{\mu}^{(0)} = \bar{\chi}$;

$\tilde{\Sigma}^{(0)} = \frac{\sum_{i=1}^n (x_i-\hat{\mu}^{(0)})(x_i-\hat{\mu}^{(0)})^T}{n}$;

répéter

  pour *i = 1 à n* **faire**

    $e_i^{(\tau-1)} = d_{\mathcal{M}}(x_i;\tilde{\mu}^{(\tau-1)},\tilde{\Sigma}^{(\tau-1)})$;

    $w_i^{(\tau-1)} = \frac{\psi(e_i^{(\tau-1)})}{e_i^{(\tau-1)}}$;

  $\tilde{\mu}^{(\tau)} = \frac{\sum_{i=1}^n w_i^{(\tau-1)} x_i}{\sum_{i=1}^n w_i^{(\tau-1)}}$;

  $\tilde{\Sigma}^{(\tau)} = \frac{\sum_{i=1}^n w_i^{(t-1)}(x_i-\tilde{\mu}^{(\tau)})(x_i-\tilde{\mu}^{(\tau)})^T}{\sum_{i=1}^n w_i^{(\tau-1)}}$;

  $\tau = \tau \leftarrow 1$;

jusqu'à *Critère d'arrêt ou de convergence*;

Comme dans le cas de l'estimation de la moyenne seule, il est nécessaire de borner le nombre d'itérations τ_{max} car on risque de sous-estimer la matrice de covariance en ne

considérant qu'un nombre trop faible de points.

4.1.3 Application au cadre de l'algorithme EM

Lors de l'étape de maximisation de l'algorithme EM pour un mélange gaussien, on rappelle que les paramètres μ_k et Σ_k pour la classe C_k sont mis à jour par les formules (Eq. (2.14) et Eq. (2.15)) :

$$\hat{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^n \hat{z}_{ik}^{(t)} x_i}{\sum_{i=1}^n \hat{z}_{ik}^{(t)}}$$

$$\hat{\Sigma}_k^{(t+1)} = \frac{\sum_{i=1}^n \hat{z}_{ik}^{(t)} (x_i - \hat{\mu}_k^{(t+1)})(x_i - \hat{\mu}_k^{(t+1)})^T}{\sum_{i=1}^n \hat{z}_{ik}^{(t)}}$$

Les probabilités a posteriori estimées $\hat{z}_{ik}^{(t)}$ sont des termes de pondération dans $[0, 1]$ des données dont nous avons montré qu'elles sont maximales pour les points aberrants. Nous proposons ainsi d'y adjoindre le terme w_i décrit précédemment pour diminuer l'influence de ces derniers. La procédure d'estimation des μ_k et Σ_k est décrite dans l'algorithme 20 où τ et t désignent respectivement l'itération courante de l'estimation et de l'algorithme EM.

Algorithme 20: Mise à jour de la moyenne et de la matrice de covariance

Données : $\chi = \{x_1, \ldots, x_n\}$, ψ la fonction d'influence du M-estimateur, $\hat{z}_{ik}^{(t)}$ une estimation de $P(C_k|x_i)$ à l'itération t de EM

Sortie : $\tilde{\mu}_k, \tilde{\Sigma}_k$ une estimation robuste de la moyenne et de la matrice de covariance

$\tau \leftarrow 0$;

$\tilde{\mu}_k^{(t+1,0)} = \frac{\sum_{i=1}^n \hat{z}_{ik}^{(t)} x_i}{\sum_{i=1}^n \hat{z}_{ik}^{(t)}}$;

$\tilde{\Sigma}_k^{(t+1,0)} = \frac{\sum_{i=1}^n \hat{z}_{ik}^{(t)} (x_i - \tilde{\mu}_k^{(t+1,0)})(x_i - \tilde{\mu}_k^{(t+1,0)})^T}{\sum_{i=1}^n \hat{z}_{ik}^{(t)}}$;

répéter

  pour *i = 1 à n* **faire**

    $e_i^{(\tau-1)} = d_{\mathcal{M}}(x_i; \tilde{\mu}_k^{(t+1,\tau-1)}, \tilde{\Sigma}_k^{(t+1,\tau-1)})$;

    $w_i^{(\tau-1)} = \frac{\psi(e_i^{(\tau-1)})}{e_i^{(\tau-1)}}$;

  $\tilde{\mu}_k^{(t+1,\tau)} = \frac{\sum_{i=1}^n w_i^{(\tau-1)} \hat{z}_{ik}^{(t)} x_i}{\sum_{i=1}^n w_i^{(\tau-1)} \hat{z}_{ik}^{(t)}}$;

  $\tilde{\Sigma}_k^{(t+1,\tau)} = \frac{\sum_{i=1}^n w_i^{(\tau-1)} \hat{z}_{ik}^{(t)} (x_i - \tilde{\mu}_k^{(t+1,\tau)})(x_i - \tilde{\mu}_k^{(t+1,\tau)})^T}{\sum_{i=1}^n w_i^{(\tau-1)} \hat{z}_{ik}^{(t)}}$;

  $\tau \leftarrow \tau + 1$;

jusqu'à *Critère d'arrêt ou de convergence*;

Il est important de noter qu'en remplaçant l'étape de maximisation de EM par cette procédure (le calcul de π_k restant inchangé), on perd la propriété initiale de l'algorithme sur la croissance monotone de $Log\mathcal{L}$ du fait de l'utilisation de l'estimation robuste récursive[13]. La phase de maximisation n'est plus strictement réalisée mais approximée. Il est donc nécessaire de changer le critère d'arrêt de l'algorithme EM traditionnellement fondé sur la stabilité de $Log\mathcal{L}$. En ce sens, nous pouvons dire que notre algorithme, qui utilise cette procédure d'estimation, n'est pas une variante de EM mais une méthode inspirée par EM. Nous reviendrons sur ce point par la suite.

4.2 Un algorithme de classification robuste

Nous avons vu dans le chapitre 2 que divers algorithmes de classification automatique et en particulier l'algorithme EM sont non robustes vis-à-vis des données aberrantes. Dans l'approche paramétrique, on peut s'interroger sur la stratégie à adopter pour robustifier un algorithme de type EM. Le chapitre 3 de ce mémoire contient déjà des éléments de réponse concernant soit l'utilisation d'estimateurs robustes et soit une modélisation robuste avec un mélange de t-distributions. Par la suite, nous proposons de compléter cette réflexion par une étude expérimentale [SJFV00b]. Avant cela, nous présentons la modélisation et la règle de rejet que nous proposons.

4.2.1 La modélisation théorique des classes

L'inclusion de la robustesse dans le modèle théorique des classes est nécessaire en l'absence d'une estimation robuste des paramètres des classes. Dans la méthode, nous proposons d'utiliser conjointement un modèle de contamination des données et une estimation robuste des paramètres de ce modèle.

Le modèle de contamination associé à une distribution de Student, appelé ϵ-contamination, est utilisé pour modéliser les données aberrantes, supposées avoir une grande variance (voir section 3.1.3.4). Rappelons que dans ce modèle, on fait l'hypothèse que les données issues d'une classe suivent la loi de probabilité (Eq. 3.23) :

$$f(x|\theta) = (1-\gamma)\mathcal{N}(x|\mu,\Sigma) + \gamma\mathcal{N}(x|\mu,\alpha\Sigma)$$

Les paramètres γ et α permettent de contrôler respectivement la proportion entre les deux composantes et l'étalement de la seconde composante en modulant sa variance.

Nous proposons d'utiliser ce modèle de contamination comme modèle théorique des

[13] C'est également le cas dans l'algorithme MCEM

classes en considérant chacune d'elles comme un mélange de deux gaussiennes :

$$f(x|\theta_j) = \underbrace{(1-\gamma_j)\mathcal{N}(x;\mu_j,\Sigma_j)}_{(A)} + \underbrace{\gamma_j\mathcal{N}(x;\mu_j,\alpha_j\Sigma_j)}_{(B)} \tag{4.3}$$

Le terme (A) est utilisé pour modéliser le noyau de la classe alors que le terme (B) permet de prendre en compte le bruit autour de la classe. On remarquera que dans (A) et (B), μ et Σ représentent les mêmes variables aléatoires qui seront estimées par des estimateurs différents selon le terme (A) ou (B). Le bruit autour d'une classe est modélisé par (B) avec une loi normale à grande variance permettant de rendre vraisemblables les données aberrantes. Cette tâche n'imcombe ainsi plus au terme (A). Cette modélisation du bruit n'implique pas nécessairement que les données bruitées suivent parfaitement ce modèle. A la limite, pour α_j très grand, (B) peut être identifié à une loi uniforme. Nous verrons par la suite que (B) peut être utilisé pour effectuer du rejet en distance.
Les paramètres γ_j et α_j permettent de contrôler respectivement la proportion entre les deux composantes et l'étalement de la seconde composante en modulant sa variance. Ainsi, γ_j est choisi entre 0 et 0.5 pour favoriser (A) et α_j est supérieur à 1. Concernant le choix du modèle de contamination, plusieurs alternatives sont possibles (Poisson, ...). Nous avons arbitrairement choisi un modèle gaussien de contamination. Cette hypothèse qui peut paraître réductrice est communément posée (Ex. : modélisation de bruits d'état et de mesure dans le filtrage de Kalman). Elle s'avère néanmoins efficace pour de nombreux problèmes réels. D'ailleurs, dans les simulations que nous allons présenter, nous avons toujours engendré un bruit additif uniforme, nous plaçant ainsi dans une situation a priori défavorable (modèle inadéquat). Les résultats obtenus ont toujours conforté ce choix. Rappelons qu'une distribution gaussienne peut être aplatie au point de tendre vers une distribution uniforme. C'est le principe du second terme (B).

4.2.2 Estimation des paramètres des classes

L'une des originalités de la méthode que nous proposons réside dans l'estimation différenciée des deux modes du modèle théorique des classes. L'idée sous-jacente est d'estimer de manière robuste les paramètres de la première composante (données issues de la classe) et d'utiliser l'estimation classique de EM pour la seconde composante. L'estimation classique, issue du principe du maximum de vraisemblance, est perturbée par les données aberrantes car elle tend à en maximiser la vraisemblance. La densité estimée par le modèle théorique de l'équation (4.3) s'écrit :

$$\hat{f}(x|\theta_k) = (1-\gamma_k)\mathcal{N}(x;\tilde{\mu}_k,\tilde{\Sigma}_k) + \gamma_k\mathcal{N}(x;\hat{\mu}_k,\alpha_k\hat{\Sigma}_k) \tag{4.4}$$

où $\tilde{\mu}_k$ et $\tilde{\Sigma}_k$ désignent les estimations robustes des moyennes et matrices de covariance. L'estimation des $\hat{\mu}_j$ et $\hat{\Sigma}_j$ repose sur les formules classiques d'EM (voir Eq. 2.14 et Eq.

Algorithme 21: Estimation différenciée de la moyenne et de la matrice de covariance

Données : $\chi = \{x_1, \ldots, x_n\}$, ψ la fonction d'influence du M-estimateur, $\hat{z}_{ik}^{(t)}$ une estimation de $P(C_k|x_i)$ à l'itération t de la boucle globale

Sortie : $\hat{\mu}_k^{(t+1)}, \hat{\Sigma}_k^{(t+1)}, \tilde{\mu}_k^{(t+1)}, \tilde{\Sigma}_k^{(t+1)}$

$\tau \leftarrow 0$;

$\tilde{\mu}_k^{(t+1,0)} = \hat{\mu}_k^{(t+1,0)} = \frac{\sum_{i=1}^n \hat{z}_{ik}^{(t)} x_i}{\sum_{i=1}^n \hat{z}_{ik}^{(t)}}$;

$\tilde{\Sigma}_k^{(t+1,0)} = \hat{\Sigma}_k^{(t+1,0)} = \frac{\sum_{i=1}^n \hat{z}_{ik}^{(t)} (x_i - \hat{\mu}_k^{(t+1,0)})(x_i - \hat{\mu}_k^{(t+1,0)})^T}{\sum_{i=1}^n \hat{z}_{ik}^{(t)}}$;

répéter

  pour *i = 1 à n* **faire**

    $e_i^{(\tau-1)} = d_{\mathcal{M}}(x_i; \tilde{\mu}_k^{(t+1,\tau-1)}, \tilde{\Sigma}_k^{(t+1,\tau-1)})$;

    $w_i^{(\tau-1)} = \frac{\psi(e_i^{(\tau-1)})}{e_i^{(\tau-1)}}$;

  $\tilde{\mu}_k^{(t+1,\tau)} = \frac{\sum_{i=1}^n w_i^{(\tau-1)} \hat{z}_{ik}^{(t)} x_i}{\sum_{i=1}^n w_i^{(\tau-1)} \hat{z}_{ik}^{(t)}}$;

  $\tilde{\Sigma}_k^{(t+1,\tau)} = \frac{\sum_{i=1}^n w_i^{(\tau-1)} \hat{z}_{ik}^{(t)} (x_i - \tilde{\mu}_k^{(t+1,\tau)})(x_i - \tilde{\mu}_k^{(t+1,\tau)})^T}{\sum_{i=1}^n w_i^{(\tau-1)} \hat{z}_{ik}^{(t)}}$;

  $\tau \leftarrow \tau + 1$;

jusqu'à *Critère d'arrêt ou de convergence*;

2.15). Les estimations robustes $\tilde{\mu}_k$ et $\tilde{\Sigma}_k$ sont initialisées à partir des estimations classiques $\hat{\mu}_j$ et $\hat{\Sigma}_j$. L'algorithme 21 récapitule ces deux estimations à partir de la procédure 20.

L'utilisation conjointe des deux estimateurs conduit à obtenir une densité $\hat{f}(x|\theta_k)$ par classe non symétrique. Dans la figure (4.2) qui illustre ce point, nous avons pris $\gamma_k = 0.5$ et $\alpha_k = 2$.

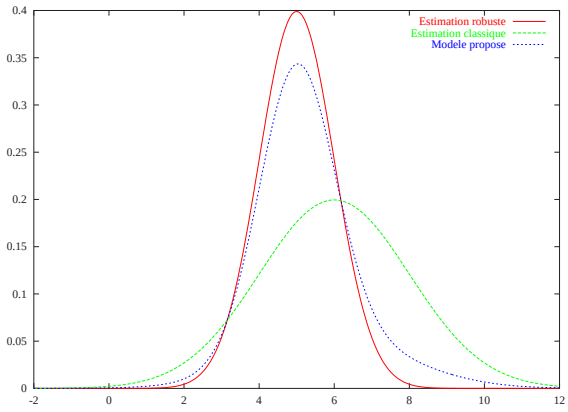


FIG. 4.2 – Densité estimée d'une classe : sous-composante robuste (A), sous-composante classique (B), bi-mode $\hat{f}(x|\theta_k)$

En pratique, si l'on dispose d'un ensemble de points χ de cardinalité n suffisante, on

peut très bien le diviser en deux sous-ensembles χ_r et χ_c ($n_r+n_c = n$) sur lesquels les deux estimations (robuste et classique) des paramètres seront réalisées sur deux échantillons différents. On évitera ainsi l'écueil d'un apprentissage biaisé.

L'expérience que nous présentons maintenant a pour objectif de valider l'algorithme proposé par rapport à sa capacité à identifier les paramètres théoriques d'un mélange simulé de données propres plus ou moins mélangées auxquelles est ajouté des données bruitées. Pour cela, nous avons généré un ensemble de 6 jeux de données mono-dimensionnelles. Chacun contient 125 points obtenus à partir de deux distributions gaussiennes (μ_1, σ_1) et (μ_2, σ_2) (50 points chacune), et d'une distribution uniforme sur $[-20; 50]$ (25 points). La première classe théorique est fixe $\mu_1 = 0$, d'écart-type constant $\sigma_1 = 4$. Chaque jeu correspond à une position différente de la deuxième classe, $\mu_2 = 0, 4, 8, 12, 16, 20$ d'écart-type constant $\sigma_2 = 2$. On fait donc varier le chevauchement des deux classes, de la confusion totale ($\mu_1 = \mu_2$) à une séparation complète ($\mu_2 >> \mu_1$). Les points de la distribution uniforme jouent le rôle des données bruitées. Ayant deux classes propres, nous avons initialisé notre algorithme pour une recherche de deux classes bi-modales ; rappelons au passage que le problème du nombre de classes n'est pas notre sujet d'étude. Différentes valeurs des paramètres α_k, γ_k et du seuil h_k du M-estimateur de Huber ont été testées indépendamment des classes en prenant $\alpha_k = \alpha$, $\gamma_k = \gamma$, $h_k = h$. La table (4.1) présente les paramètres obtenus sur les 6 jeux de données que l'on illustre sur la figure (4.3).

	$\mu_2 = 0$	$\mu_2 = 4$	$\mu_2 = 8$	$\mu_2 = 12$	$\mu_2 = 16$	$\mu_2 = 20$
$\hat{\pi}_1$	0.3970	0.1992	0.2843	0.4578	0.4882	0.4857
$\tilde{\mu}_1$	-0.3900	-1.4388	-0.2418	1.2168	1.3465	1.4023
$\hat{\mu}_1$	0.1892	-2.4175	-0.8999	4.3792	4.5482	4.7716
$\tilde{\sigma}_1$	1.7828	1.4266	1.7487	2.5697	2.7840	2.9033
$\hat{\sigma}_1$	4.7838	3.4105	3.8211	10.6871	10.8050	10.7485
$\hat{\pi}_2$	0.6030	0.8008	0.7157	0.5422	0.5118	0.5143
$\tilde{\mu}_2$	2.7459	3.7329	7.4112	12.3920	16.4637	20.2030
$\hat{\mu}_2$	8.0018	8.2432	11.2718	14.0358	17.6017	20.0804
$\tilde{\sigma}_2$	4.5367	2.002	2.4416	2.0935	2.0888	1.8755
$\hat{\sigma}_2$	3.3502	11.2916	10.8468	9.4650	8.6107	8.8806

TAB. 4.1 – *OneDim :* Paramètres estimés sur les 6 jeux de données

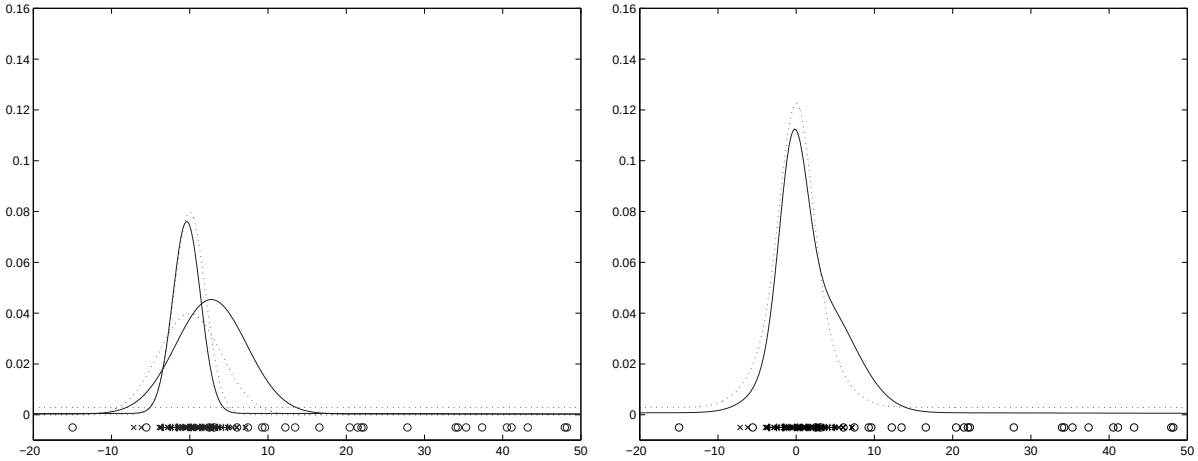
A chaque niveau de la figure correspond un jeu de données. Sur la partie gauche sont représentées les densités théoriques $f(x|C_1)$ et $f(x|C_2)$ (en pointillé), ainsi que les densités estimées (en trait plein) $\hat{f}(x|C_1)$ et $\hat{f}(x|C_2)$ des deux classes ; les densités estimées sont obtenues en reconstruisant le bi-mode (voir Eq. (4.4)). En surimpression, les points ont été affichés de manière symbolique : $+$ pour C_1, $\times$ pour C_2 et o pour le bruit additif.

Afin de ne pas nuire à la lisibilité de la figure, les 4 sous-composantes (2 par classe) n'ont pas été représentées. Sur la partie droite, la même légende sert à la représentation des densités mélange estimées $\hat{f}(x) = \hat{\pi}_1\hat{f}(x|C_1) + \hat{\pi}_2\hat{f}(x|C_2)$ (en pointillé) et théoriques $f(x) = \pi_1 f(x|C_1) + \pi_2 f(x|C_2)$ (en trait plein).

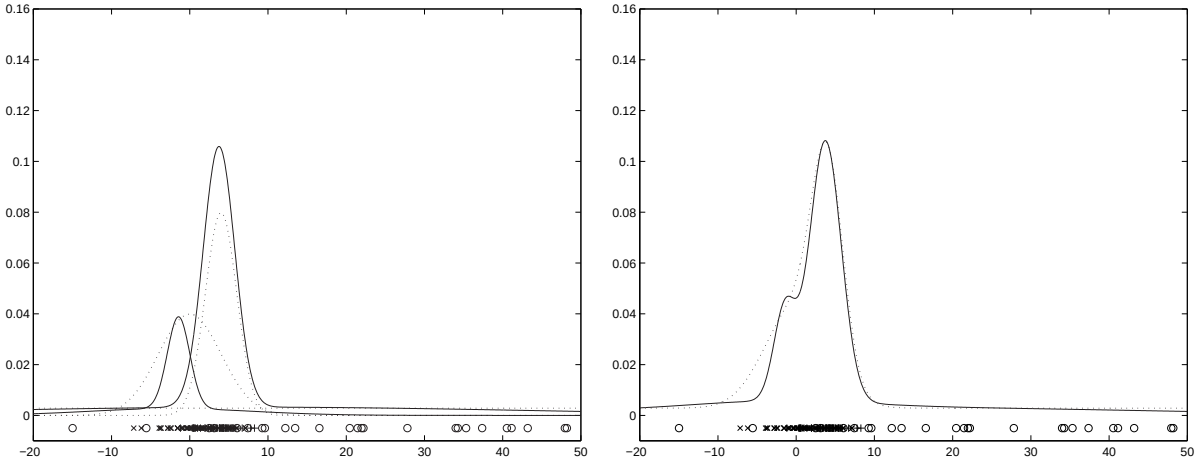
Comme on pouvait s'y attendre, les premiers exemples contenant un fort chevauchement ($\mu_2 = 0,4$) sont difficiles à traiter. Nous pouvons voir dans la figure (4.3) que la densité mélange théorique (en pointillé sur la partie droite) ne laisse pas apparaître la présence de deux modes. Dans ces conditions, l'estimation de la densité favorise généralement l'une des deux classes. Ce phénomène s'explique par le fait que la contrainte de normalisation des probabilités a posteriori force les classes détectées à être disjointes (voir Fig. (3.2)). On constate ainsi une prépondérance de la classe C_2 pour ces deux jeux de données ($\hat{\pi}_2 = 0.603$ et 0.8008). Cette prédominance force l'un des deux modes à être décalé. Du fait de la présence d'un plus grand nombre de points de la densité uniforme du coté droit, ce décalage s'opère vers la droite. Notons qu'il est nettement moins important dans le cas de l'estimation robuste (voir les valeurs de $\tilde{\mu}_2$ et $\hat{\mu}_2$). Pour la même raison, l'écart-type de la classe C_1 est sous-estimé ($\tilde{\sigma_1} = 1.7828$ et 1.4266 au lieu de 4). Ainsi, pour $\mu_2 = 0$, l'écart-type robuste $\tilde{\sigma_2}$ de la classe C_2 est proche de 4 qui est l'écart-type théorique le plus grand de deux classes ($\sigma_1 = 4, \sigma_2 = 2$).

Au fur et à mesure que la séparabilité des classes des classes augmente, la prépondérance d'une des deux classes tend à disparaître, même si celle-ci reste marquée dans le cas $\mu_2 = 8$ ($\hat{\pi}_2 = 0.7157$). La présence de plus de points de bruit uniforme sur la droite conduit à estimer $\hat{\pi}_2$ légèrement supérieur à $\hat{\pi}_1$. Concernant les moyennes estimées $\tilde{\mu}_2$ et $\hat{\mu}_2$ pour C_2, on observe une supériorité de l'estimation robuste sur l'estimation classique. La différence entre les deux types d'estimations tend à disparaître lorsque les classes sont complètement séparées ($\mu_2 = 20$). Pour la classe C_1, on peut noter qu'il subsiste un décalage systématique vers la droite. On peut expliquer ce phénomène de la manière suivante. La moyenne expérimentale des 50 seuls points issus de la composante C_1 engendrée est de 0.9 alors que leur médiane vaut 1.07. Le léger biais constaté dans la génération des données se retrouve dans l'estimation car le nombre de points est insuffisant pour être proche des paramètres théoriques. Au fur et à mesure que les classes se séparent, on observe aussi que l'écart-type robuste $\tilde{\sigma}_1$ correspondant à C_1 est systématiquement sous-estimé. On peut supposer que le choix du paramètre commun h ($h_1 = h_2$) a privilégié le mode correspondant à C_2 dont la variance est plus faible. De ce fait, les données de la première composante ont été trop filtrées par le M-estimateur. Cela illustre la nécessité de choisir de manière indépendante, classe par classe, le seuil du M-estimateur. Du fait des données de bruit, les écart-types des classes sont très sur-estimés par les estimateurs non robustes $\hat{\sigma}_1$ et $\hat{\sigma}_2$. Cela confirme que l'estimation classique des écart-types est influencée à la fois par les données propres et les données bruitées qui forcent l'aplatissement de la loi normale.

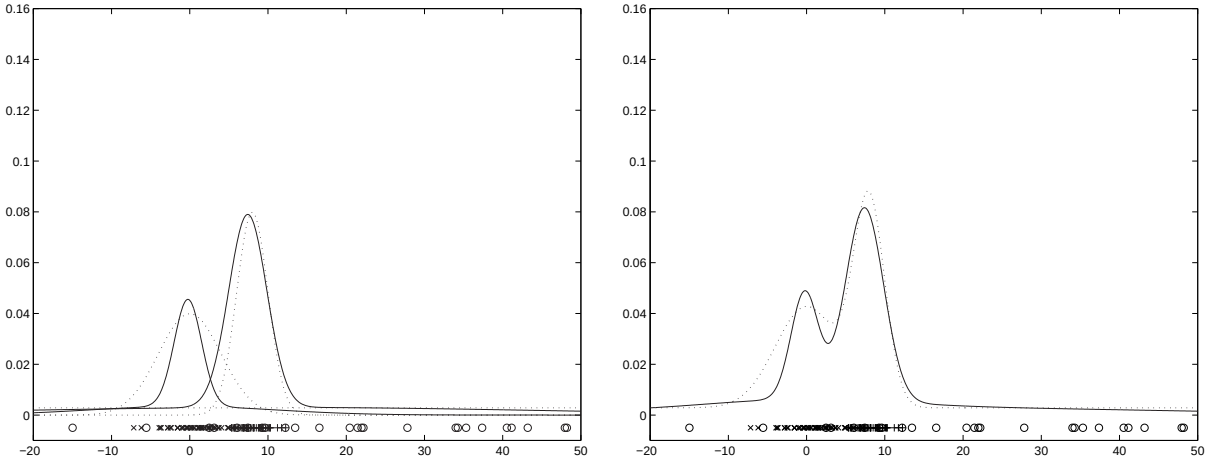
$\mu_2 = 0$



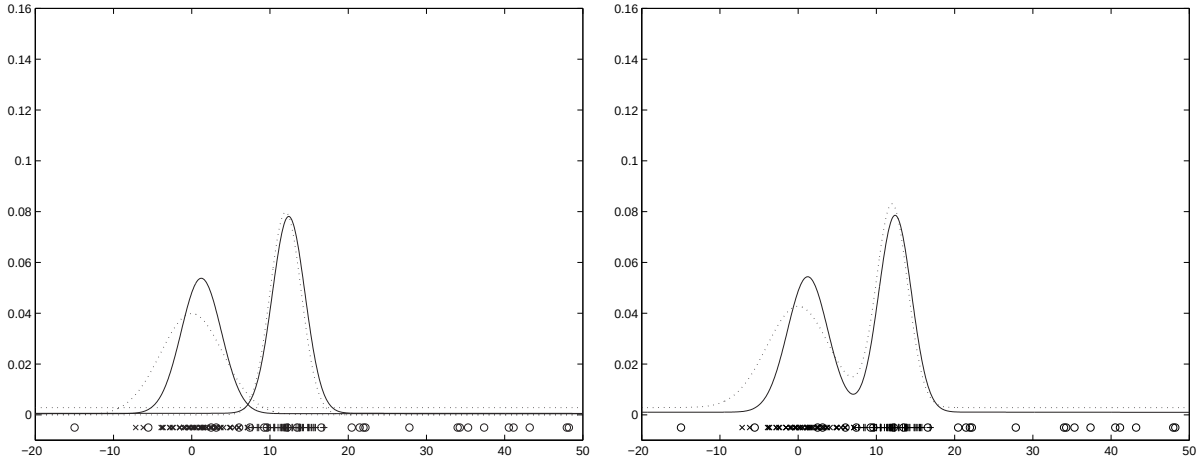
$\mu_2 = 4$



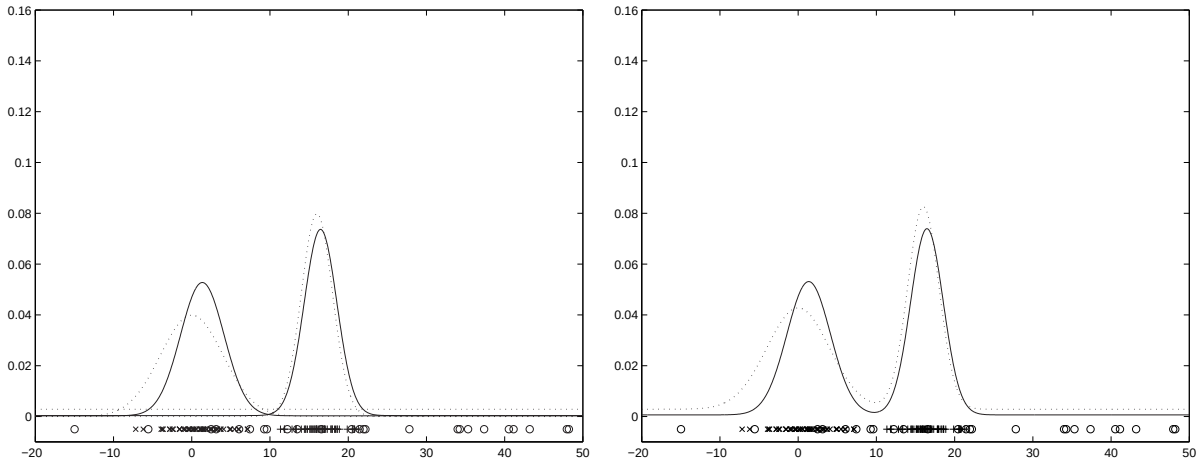
$\mu_2 = 8$



$\mu_2 = 12$



$\mu_2 = 16$



$\mu_2 = 20$

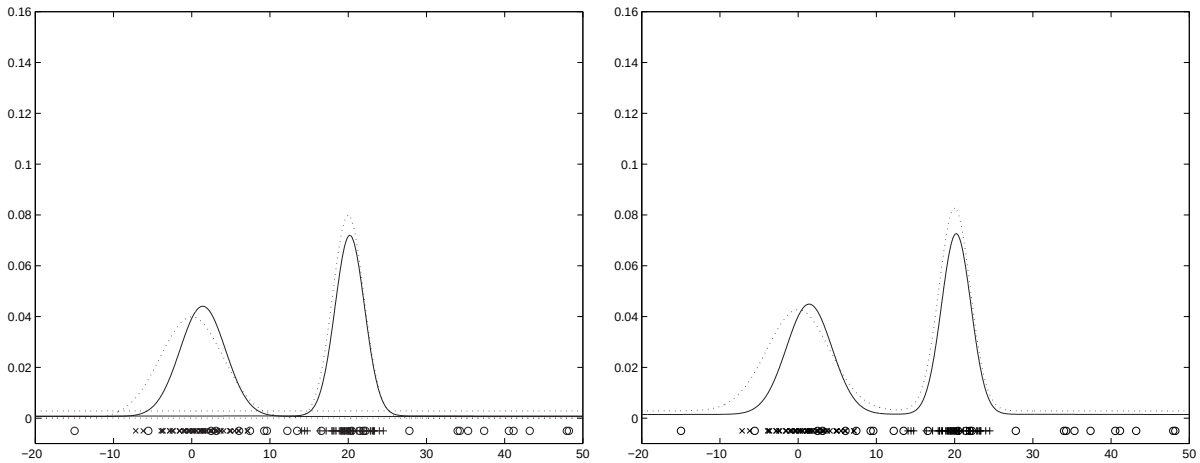


FIG. 4.3 – A gauche : la densité par composante - A droite, la densité mélange

Les résultats obtenus par notre méthode sont satisfaisants sans toutefois que le modèle de bruit additif des données (bruit uniforme) et le modèle de contamination soient identiques. On peut supposer que si tel était le cas, les résultats n'en seraient que plus favorables.

4.2.3 Règle de décision et rejet

Lors de la phase de validation d'un discriminateur, on est amené à classer un ensemble de données de test. Certains algorithmes tels que CEM nécessitent également d'effectuer une partition dure des données lors de l'apprentissage. Ces deux classements peuvent s'avérer problématiques sur certaines données lorsqu'on suppose que l'on va commettre une erreur d'affectation. Il est possible alors de refuser l'affectation et, dans ce cas, les points seront qualifiés de points rejetés. Il existe deux causes possibles de rejet :

- le rejet en distance que l'on effectue lorsqu'un point est trop éloigné du prototype de la classe. Ce cas correspond typiquement au rejet des points aberrants [DM93].
- le rejet en ambiguïté qui intervient lorsque un point est proche d'une frontière de décision [Ha97]. Le degré d'appartenance ou la probabilité a posteriori de ce point à l'une des classes n'est pas nettement majoritaire. Dans le cas limite, les degrés d'appartenance d'un point appartenant à une frontière de décision sont égaux pour plusieurs classes.

Pour une présentation unifiée des discriminateurs avec double option de rejet (distance, ambiguïté) dans les cadre flous, probabilistes et possibilistes, le lecteur peut se référer à [Fré98b]. Ces deux types de rejet peuvent être effectués pendant ou après la phase d'apprentissage. Il est difficile de parler de rejet en amont d'une classification : la recherche des zones d'ambiguïté est une problématique similaire à celle de la recherche de frontières de décision. Concernant le rejet en distance, il est préférable de parler de pré-traitement ou de filtrage des données aberrantes.

On peut se demander s'il est préférable d'effectuer le rejet avant, pendant ou après la classification. A ce propos, Huber constate expérimentalement que [Hub81, p. 5] :

> "It is an empirical fact that the best rejection procedures do not quite reach the performance of the best robust procedures. The latter apparently are superior because they can make smooth transition between full acceptance and full rejection of an observation."

Celui-ci fait référence ici aux méthodes d'estimation rejetant les points que l'on ne désire pas prendre en compte. On peut généraliser son propos aux algorithmes de classification utilisant une partition stricte des données, au lieu d'une partition floue (Ex. HCM/FCM).

Pour le modèle théorique que nous proposons, le rejet en distance des points aberrants peut être effectué naturellement lors de la phase de d'affectation finale des points aux classes [SJFV00a]. Pour cela, on crée une (c+1)-ème classe, appelée classe de rejet, dans laquelle on regroupe tous les seconds termes (B) du mélange intra-classe. Cela revient ainsi à redéfinir le mélange sous la forme :

$$\begin{aligned}
\hat{f}(x) &\equiv f(x;\hat{\Theta}) \\
&= \sum_{k=1}^{c} \hat{\pi}_k f(x;\hat{\Theta}_k) \\
&= \sum_{k=1}^{c} \hat{\pi}_k \left[(1-\gamma_k)\,\mathcal{N}(x;\tilde{\mu}_k,\tilde{\Sigma}_k) + \gamma_k\,\mathcal{N}(x;\hat{\mu_k},\alpha_k\hat{\Sigma_k})\right] \\
&= \underbrace{\sum_{k=1}^{c} \hat{\pi}_k \left[(1-\gamma_k)\,\mathcal{N}(x;\tilde{\mu}_k,\tilde{\Sigma}_k)\right]}_{\text{c classes}} + \underbrace{\sum_{k=1}^{c} \hat{\pi}_k \left[\gamma_k\,\mathcal{N}(x;\hat{\mu_k},\alpha_k\hat{\Sigma_k})\right]}_{\text{classe de rejet}}
\end{aligned} \tag{4.5}$$

L'affectation s'effectue ensuite suivant le critère du Maximum A Posteriori (MAP) parmi ces c+1 classes :

$$\hat{P}(C_k|x) = \frac{\hat{\pi}_k(1-\gamma_k)\,\mathcal{N}(x;\tilde{\mu}_k,\tilde{\Sigma}_k)}{\hat{f}(x)} \tag{4.6}$$

$$\hat{P}(Rejet|x) = \frac{\sum_{k=1}^{c} \hat{\pi}_k\gamma_k\,\mathcal{N}(x;\hat{\mu_k},\alpha_k\hat{\Sigma_k})}{\hat{f}(x)} \tag{4.7}$$

4.2.4 Choix des paramètres , Compromis Erreur/Rejet

Lorsque l'on effectue du rejet, il est généralement nécessaire de fixer un ou plusieurs seuils paramètrant le taux de rejet (Ex. : Distance à la classe de bruit dans FNC, niveau de l'α-coupe). Bien évidemment, rejeter des points implique une diminution à la fois du taux d'erreur mais également du taux de succès : on s'évite ainsi de classer à tort comme à raison. Dans le cas limite où l'on rejette tous les points sauf un, on obtient un taux de succès de 100% (et un taux d'erreur de 0%) en classant correctement le seul point non rejeté.

Dans certaines applications sensibles (Ex. : Sécurité nucléaire, militaire, médicale), il est souhaitable plus qu'ailleurs de limiter le nombre d'erreurs commises car le coût associé à un mauvais classement est grand. On est donc amené à effectuer un compromis entre l'erreur et le rejet.

Dans notre modèle, les paramètres γ_k et α_k contrôlent la proportion de bruit considéré autour d'une classe. Le taux de rejet est élevé lorsque ces deux valeurs sont grandes. γ_k

étant une probabilité a priori, il convient de le choisir dans $[0;1]$. Cependant, il semble naturel, d'après la sémantique du modèle, de prendre une valeur inférieure à 0.5. Pour la même raison, α_k doit être supérieur à 1. Il est difficile de choisir manuellement ces paramètres pour chacune des classes sans connaissance a priori sur les classes. Nous avons ainsi opter pour des valeurs identiques pour chacune des classes : $\gamma_k = \gamma$ et $\alpha_k = \alpha$.

Pour avoir une intuition sur de bonnes valeurs pour α et γ par rapport au domaine d'application, nous pouvons utiliser la phase de test de la classification. Pour un paramétrage Γ donné, on peut définir par exemple la qualité de Γ comme une combinaison linéaire des probabilités P_c de succès, P_e d'erreur et P_r de rejet estimées :

$$C(\Gamma) = C_c P_c - C_e P_e - C_r P_r \tag{4.8}$$

où C_c, C_e et C_r désignent respectivement les coûts associés au succès, à l'erreur et au rejet (C_c, C_e et C_r sont positifs). Ce critère $C(\Gamma)$ étant à maximiser, majorer par exemple C_e revient sélectionner le paramétrage Γ qui minimise l'erreur. Cela se produit généralement pour des valeurs $\gamma > 4$ et α entre 0.25 et 0.5 permettant de rejeter un grand nombre de points.

Dans la plupart des expérimentations que nous présentons par la suite, nous avons fixé les coûts aux valeurs $C_c = 1, C_e = 2, C_r = 2$.

4.2.5 Expérimentations

Nous présentons ici deux types d'expérimentations. Le premier consiste en une étude comparative de notre modèle par rapport à d'autres modélisations du bruit. Il s'agit essentiellement de comparer le qualité de l'estimation produite. Dans un second temps, nous présentons et comparons les résultats que nous avons obtenus sur divers jeux de données artificiels et réels.

4.2.5.1 Evaluation comparative du modèle proposé

L'objectif de cette étude est de comparer le modèle proposé par rapport à des modèles où l'on effectue soit une modélisation explicite du bruit, soit une estimation robuste, soit encore les deux à la fois. Ainsi, nous avons considéré 5 méthodes de classification différentes.

4.2.5.1.1 Les différents modèles testés

1. c composantes gaussiennes dont les paramètres sont estimés par l'algorithme EM (Eq. 2.13) – Eq. (2.15) :

$$f(x;\Theta) = \sum_{k=1}^{c} \pi_k \, \mathcal{N}(x;\mu_k,\Sigma_k) \tag{4.9}$$

2. c composantes gaussiennes et une composante uniforme

$$f(x;\Theta) = \gamma\, \mathcal{U}(x;H) + \sum_{k=1}^{c} \pi_k\, \mathcal{N}(x;\mu_k,\Sigma_k) \tag{4.10}$$

   où γ est fixé par l'utilisateur

3. c composantes gaussiennes dont les paramètres sont estimés par l'algorithme 19 :

$$f(x;\Theta) = \sum_{k=1}^{c} \pi_k\, \mathcal{N}(x;\mu_k,\Sigma_k) \tag{4.11}$$

4. c composantes gaussiennes dont les paramètres sont estimés par l'algorithme 19 et et une composante uniforme

$$f(x;\Theta) = \gamma\, \mathcal{U}(x;H) + \sum_{k=1}^{c} \pi_k\, \mathcal{N}(x;\mu_k,\Sigma_k) \tag{4.12}$$

   où γ est fixé par l'utilisateur

5. Notre modèle : c composantes bi-modales estimées par l'algorithme 21

$$\hat{f}(x|\theta_k) = (1-\gamma_k)\mathcal{N}(x;\tilde{\mu}_k,\tilde{\Sigma}_k) + \gamma_k\mathcal{N}(x;\hat{\mu}_k,\alpha_k\hat{\Sigma}_k) \tag{4.13}$$

   où les $\gamma_k = \gamma$ et $\alpha_k = \alpha$ sont fixés par l'utilisateur.

4.2.5.1.2 Le protocole expérimental

Afin de tester les performances respectives de ces différents modèles, nous avons produit deux jeux de données artificiels (nommés *Easy* et *Dif*). Chacun d'eux contient 3 composantes gaussiennes plus une composante uniforme constituant un bruit additionnel. Dans *Easy*, les classes sont relativement bien séparées alors que dans *Dif* les trois classes se chevauchent (voir Fig. 4.4). Les taux de contamination (rapport du nombre de points uniformément distribués sur le nombre total de points engendrés) sont respectivement de 30% et 16.67%. La table 4.2 récapitule les différents paramètres théoriques des classes. Les coefficients de corrélation r_1, r_2 et r_3, reportés en supplément, décrivent l'orientation des classes.

Le jeu de données *Dif* semble particulièrement difficile car la zone de chevauchement des classes correspond à un mode de la densité (voir Fig. 4.5).

Comme dans la plupart des algorithmes de partitionnement, il est nécessaire de fixer le nombre de classes de classes recherchées. Nous avons choisi de ne pas considérer ce problème dans cette étude et avons fixé c à 3. On peut tout de même noter que divers critères d'information indiquent qu'il convient de rechercher 4 classes pour *Easy* et 3 ou 4 pour *Dif*. Nous avons reporté sur la figure (4.6) les résultats obtenus par les critères BIC,

	Easy	*Dif*
μ_1, Σ_1, r_1, n_1	$\begin{bmatrix}7\\9\end{bmatrix}$, $\begin{bmatrix}3 & -2\\-2 & 1.5\end{bmatrix}$, $-\frac{2\sqrt{2}}{3}$, 100	$\begin{bmatrix}3\\5\end{bmatrix}$, $\begin{bmatrix}1 & 0\\0 & 3\end{bmatrix}$, 0, 200
μ_2, Σ_2, r_2, n_2	$\begin{bmatrix}6\\3\end{bmatrix}$, $\begin{bmatrix}3 & 0\\0 & 1\end{bmatrix}$, 0, 150	$\begin{bmatrix}6\\9\end{bmatrix}$, $\begin{bmatrix}2 & 1.5\\1.5 & 2\end{bmatrix}$, $\frac{3}{4}$, 150
μ_3, Σ_3, r_3, n_3	$\begin{bmatrix}3\\7\end{bmatrix}$, $\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$, 0, 100	$\begin{bmatrix}6\\5\end{bmatrix}$, $\begin{bmatrix}3.25 & -3\\-3 & 3.25\end{bmatrix}$, $-\frac{12}{13}$, 150
Uniforme, n_*	$[0,12]\times[0,12]$, 150	$[0,12]\times[0,12]$,100

TAB. 4.2 – Paramètres théoriques des jeux de données *Easy* et *Dif*

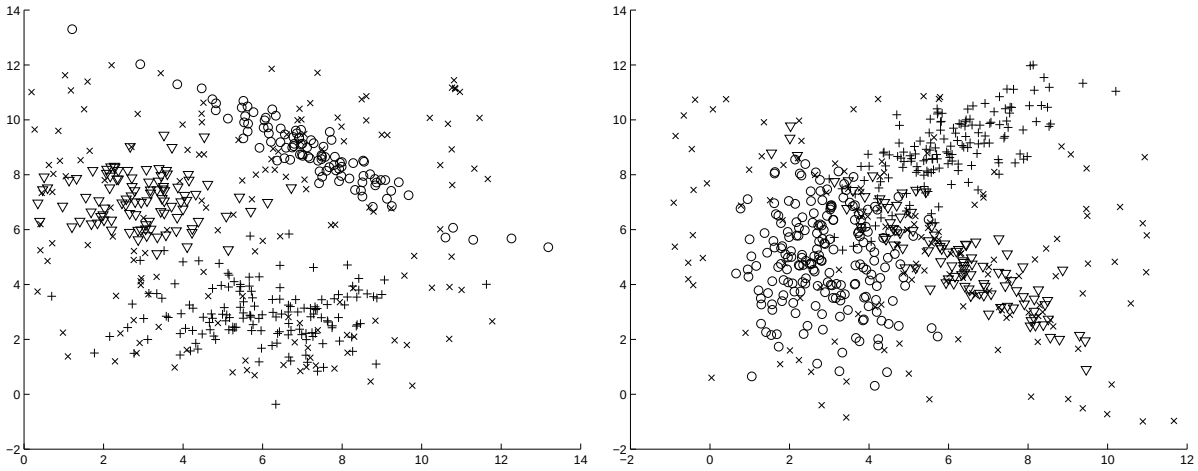


FIG. 4.4 – Les jeux de données *Easy* (à gauche) et *Dif* (à droite) ; $C_1(o), C_2(+), C_3(\triangledown), C_4(x)$

ICL pour des modèles de classes gaussiennes non contraintes. Le minimum de chaque critère donne le nombre de composantes à sélectionner. Sur *Easy*, les deux critères indiquent clairement 4 composantes alors le choix est partagé entre 3 et 4 pour *Dif*. La figure (4.7) montre que la classe supplémentaire est, dans les deux cas, composée de points de bruit.

Afin d'être insensible à l'initialisation des paramètres des classes, nous avons effectué, pour chacun des modèles testés, 50 initialisations identiques aléatoires : les centres initiaux des classes sont choisis au hasard parmi les points et les matrices de covariance sont initialisées à l'identité. Le M-estimateur retenu pour les modèles avec une estimation robuste est ici celui de Cauchy dont nous avons fait varier le paramètre. Différentes valeurs des paramètres α_k et γ_k ont été testées.

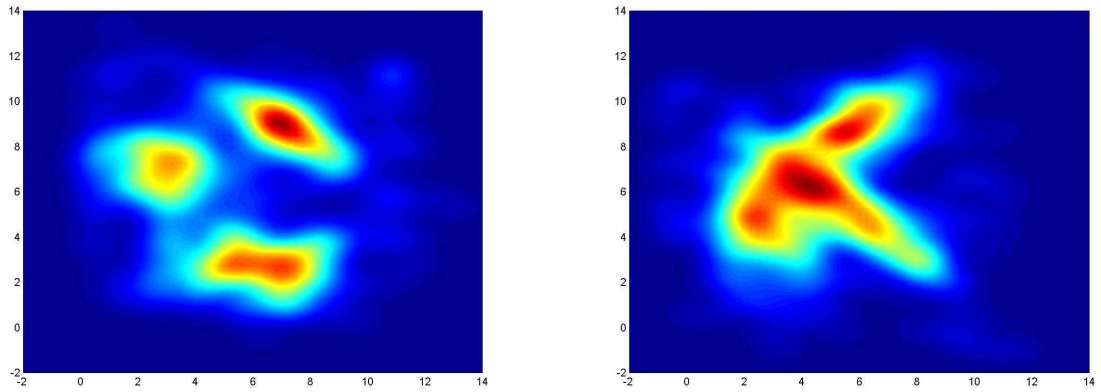


FIG. 4.5 – Densité estimée par la méthode des noyaux de Parzen pour *Easy* (à gauche) et *Dif* (à droite)

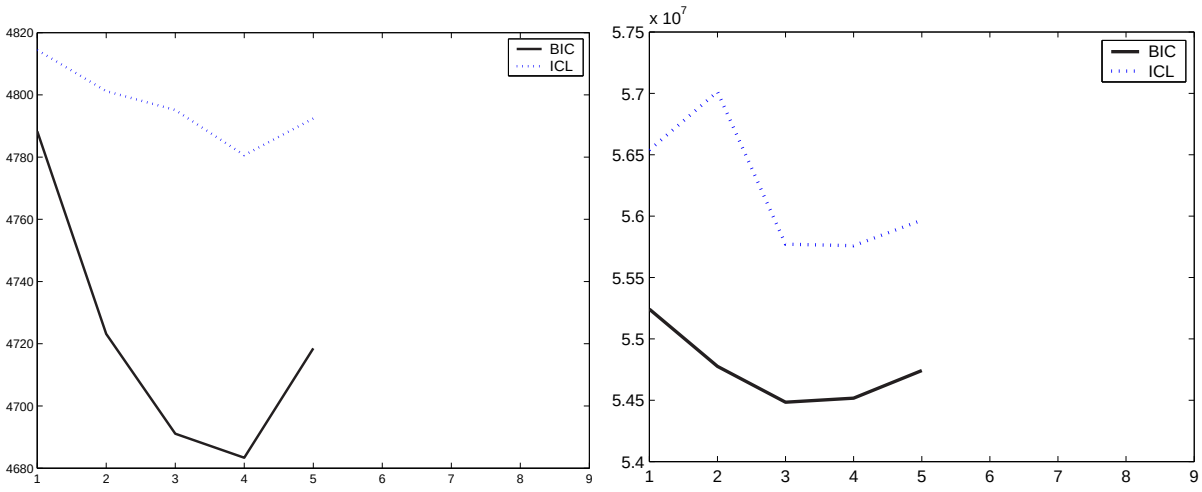


FIG. 4.6 – Critères ICL et BIC pour *Easy* (à gauche) et *Dif* (à droite)

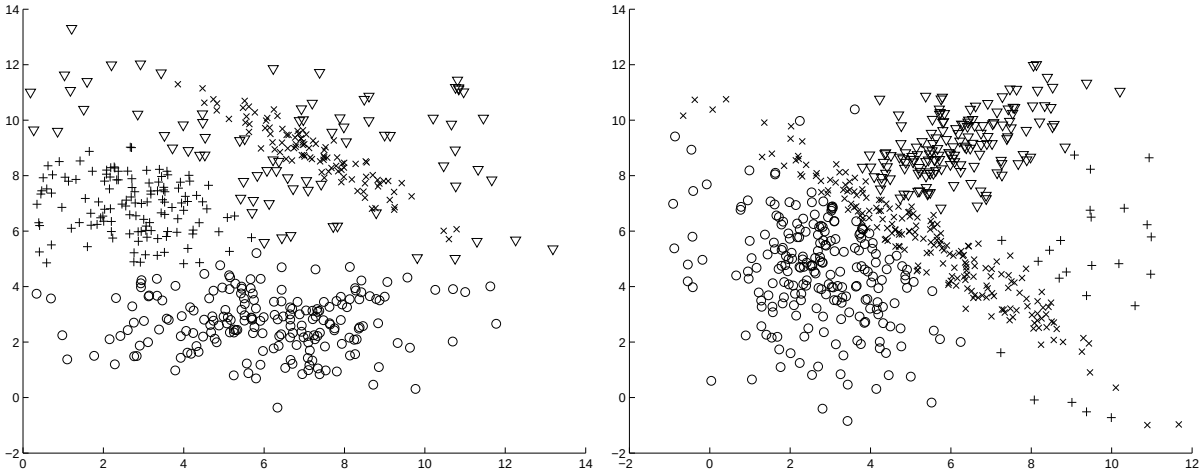


FIG. 4.7 – *Easy* (à gauche) et *Dif* (à droite) pour 4 classes

	Modèle #1	Modèle #2	Modèle #3	Modèle #4	Modèle#5
μ_1	$\begin{pmatrix} 7.07 \\ 8.91 \end{pmatrix}$	$\begin{pmatrix} 7.18 \\ 8.87 \end{pmatrix}$	$\begin{pmatrix} 7.17 \\ 8.86 \end{pmatrix}$	$\begin{pmatrix} 7.14 \\ 8.84 \end{pmatrix}$	$\begin{pmatrix} 7.16 \\ 8.86 \end{pmatrix}$
Σ_1	$\begin{pmatrix} 5.32 & -1.69 \\ -1.69 & 2.28 \end{pmatrix}$	$\begin{pmatrix} 2.49 & -1.59 \\ -1.59 & 1.56 \end{pmatrix}$	$\begin{pmatrix} 0.89 & -0.63 \\ -0.63 & 0.59 \end{pmatrix}$	$\begin{pmatrix} 1.72 & -1.2 \\ -1.2 & 1.17 \end{pmatrix}$	$\begin{pmatrix} 2.77 & -1.36 \\ -1.36 & 1.47 \end{pmatrix}$
r_1	-0.48	-0.81	-0.87	-0.87	-0.67
μ_2	$\begin{pmatrix} 6.07 \\ 2.74 \end{pmatrix}$	$\begin{pmatrix} 6.12 \\ 2.73 \end{pmatrix}$	$\begin{pmatrix} 6.21 \\ 2.73 \end{pmatrix}$	$\begin{pmatrix} 6.09 \\ 2.73 \end{pmatrix}$	$\begin{pmatrix} 6.09 \\ 2.73 \end{pmatrix}$
Σ_2	$\begin{pmatrix} 4.46 & -0.01 \\ -0.01 & 1.13 \end{pmatrix}$	$\begin{pmatrix} 4.06 & 0.08 \\ 0.08 & 1.04 \end{pmatrix}$	$\begin{pmatrix} 1.77 & -0.01 \\ -0.01 & 0.41 \end{pmatrix}$	$\begin{pmatrix} 3.31 & 0.00 \\ 0.00 & 0.89 \end{pmatrix}$	$\begin{pmatrix} 3.45 & -0.00 \\ -0.00 & 0.90 \end{pmatrix}$
r_2	-0.00	0.04	-0.01	-0.00	-0.00
μ_3	$\begin{pmatrix} 2.77 \\ 6.90 \end{pmatrix}$	$\begin{pmatrix} 2.98 \\ 7.00 \end{pmatrix}$	$\begin{pmatrix} 2.99 \\ 7.06 \end{pmatrix}$	$\begin{pmatrix} 2.97 \\ 7.03 \end{pmatrix}$	$\begin{pmatrix} 2.86 \\ 6.99 \end{pmatrix}$
Σ_3	$\begin{pmatrix} 1.72 & -0.38 \\ -0.38 & 1.26 \end{pmatrix}$	$\begin{pmatrix} 2.05 & -0.37 \\ -0.37 & 1.66 \end{pmatrix}$	$\begin{pmatrix} 1.03 & -0.08 \\ -0.08 & 0.76 \end{pmatrix}$	$\begin{pmatrix} 1.78 & -0.2 \\ -0.2 & 1.22 \end{pmatrix}$	$\begin{pmatrix} 1.56 & -0.31 \\ -0.31 & 1.12 \end{pmatrix}$
r_3	-0.26	-0.20	-0.09	-0.13	-0.23
Erreur	3.43%	3.43%	2.%	3.43%	2.86%

TAB. 4.3 – Résultats sur *Easy* : paramètres estimés et erreur

4.2.5.1.3 Résultats

Le premier aspect considéré est la qualité de l'estimation produite par chacun des modèles (erreur d'estimation), i.e. leur capacité à identifier les paramètres théoriques des classes. Pour cela, nous avons retenu les meilleurs résultats en resubstitution sur les c premières classes pour chacun d'eux (la classification des points issus de la classe de bruit n'est pas prise en compte). Les tables (4.3) et (4.4) récapitulent les différents résultats obtenus sur *Easy* et *Dif*.

Comme on pouvait s'y attendre tous les modèles identifient correctement les moyennes théoriques des classes sur le jeu *Easy*. De plus, les taux d'erreurs, obtenus par resubsti-

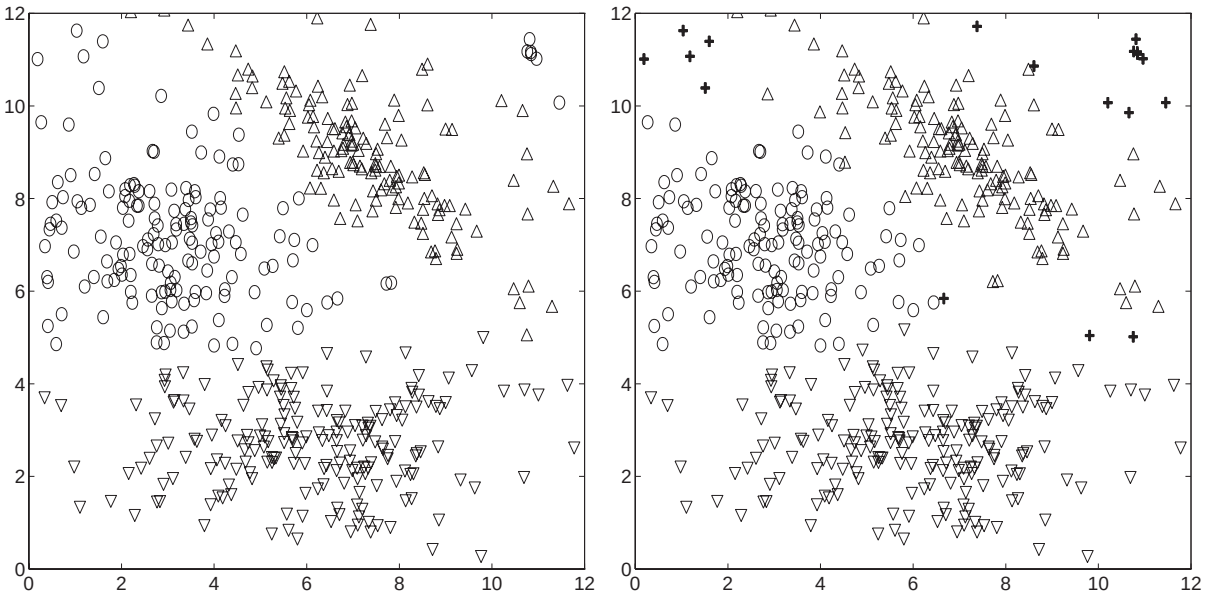


FIG. 4.8 – *Easy* : Partitionnement obtenu par les modèles #3 (à gauche) et notre modèle #5 (à droite)- les + sont les points rejetés

tution sur les classes gaussiennes, sont faibles pour tous les modèles. L'erreur minimale pour les modèles #2,#4 et #5 a été obtenue pour des valeurs faibles des γ et α, ce qui indique une faible modélisation des points de bruit. Il est en de même pour le paramètre de Cauchy qui croît inversement proportionnellement au nombre de points éliminés dans l'estimation robuste. En revanche, nous pouvons voir que l'estimation robuste (modèle #3,#4 et #5) permet d'améliorer l'estimation de la taille et de l'orientation des classes caractérisées par les matrices de covariance et par les coefficients de corrélation. Cela peut s'expliquer par le fait que les points de la distribution uniforme autour des classes amènent à sur-estimer leurs tailles. Le meilleur résultat, obtenu par le modèle #3 est montré sur la partie gauche de la figure (4.8). On remarquera dans la partie supérieure droite de cette figure que certains points de bruit ont été classés dans la classe des ronds du fait de l'usage de la distance de Mahalanobis. Sur la partie droite de la figure, on voit que notre modèle a correctement rejeté ces points. Les erreurs d'affectation se situent entre les frontières de décision des classes $(C_1(\triangle) - C_3(o))$ et $(C_2(\triangledown) - C_3(o))$ (10 points).

Pour le jeu de données *Dif* où les classes se chevauchent plus, les paramètres différent nettement pour chaque modèle en particulier pour ceux de la classe 3 (voir table (4.4)). Nous pouvons remarquer en particulier que les matrices de covariance sont largement sur-estimées dans le cas où l'on n'utilise pas l'estimation robuste (modèles #1 et #2). La figure (4.10) montre que l'algorithme EM cherche également à maximiser la vraisemblance des données aberrantes. On constate ainsi que la valeur optimale du seuil de Cauchy (pour les

	Modèle #1	Modèle #2	Modèle #3	Modèle #4	Modèle #5
μ_1	$\begin{pmatrix} 3.97 \\ 5.06 \end{pmatrix}$	$\begin{pmatrix} 2.84 \\ 4.65 \end{pmatrix}$	$\begin{pmatrix} 2.74 \\ 4.57 \end{pmatrix}$	$\begin{pmatrix} 3.02 \\ 5.20 \end{pmatrix}$	$\begin{pmatrix} 2.95 \\ 5.11 \end{pmatrix}$
Σ_1	$\begin{pmatrix} 5.60 & 0.03 \\ 0.03 & 4.54 \end{pmatrix}$	$\begin{pmatrix} 1.45 & -0.38 \\ -0.38 & 3.14 \end{pmatrix}$	$\begin{pmatrix} 0.96 & -0.37 \\ -0.37 & 1.85 \end{pmatrix}$	$\begin{pmatrix} 1.26 & -0.10 \\ -0.10 & 3.38 \end{pmatrix}$	$\begin{pmatrix} 1.21 & -0.17 \\ -0.17 & 3.06 \end{pmatrix}$
r_1	0.01	-0.18	-0.28	-0.05	-0.09
μ_2	$\begin{pmatrix} 6.06 \\ 9.13 \end{pmatrix}$	$\begin{pmatrix} 5.99 \\ 8.94 \end{pmatrix}$	$\begin{pmatrix} 6.03 \\ 9.04 \end{pmatrix}$	$\begin{pmatrix} 6.09 \\ 9.05 \end{pmatrix}$	$\begin{pmatrix} 6.10 \\ 9.06 \end{pmatrix}$
Σ_2	$\begin{pmatrix} 2.02 & 1.19 \\ 1.19 & 1.41 \end{pmatrix}$	$\begin{pmatrix} 1.70 & 0.95 \\ 0.95 & 1.40 \end{pmatrix}$	$\begin{pmatrix} 1.07 & 0.50 \\ 0.50 & 0.81 \end{pmatrix}$	$\begin{pmatrix} 1.29 & 0.69 \\ 0.69 & 1.07 \end{pmatrix}$	$\begin{pmatrix} 1.24 & 0.56 \\ 0.56 & 0.95 \end{pmatrix}$
r_2	0.70	0.61	0.53	0.59	0.52
μ_3	$\begin{pmatrix} 5.68 \\ 5.17 \end{pmatrix}$	$\begin{pmatrix} 5.61 \\ 5.30 \end{pmatrix}$	$\begin{pmatrix} 5.45 \\ 5.45 \end{pmatrix}$	$\begin{pmatrix} 6.40 \\ 4.65 \end{pmatrix}$	$\begin{pmatrix} 6.30 \\ 4.74 \end{pmatrix}$
Σ_3	$\begin{pmatrix} 5.58 & -5.4 \\ -5.4 & 5.53 \end{pmatrix}$	$\begin{pmatrix} 3.80 & -3.37 \\ -3.37 & 3.45 \end{pmatrix}$	$\begin{pmatrix} 3.00 & -2.65 \\ -2.65 & 2.70 \end{pmatrix}$	$\begin{pmatrix} 1.90 & -1.82 \\ -1.82 & 2.06 \end{pmatrix}$	$\begin{pmatrix} 2.13 & -2.01 \\ -2.01 & 2.27 \end{pmatrix}$
r_3	-0.96	-0.93	-0.93	-0.92	-0.91
Erreur	15.33%	12.67%	14.22%	10.67%	11.33%

TAB. 4.4 – Résultats sur *Dif* : paramètres estimés et erreur

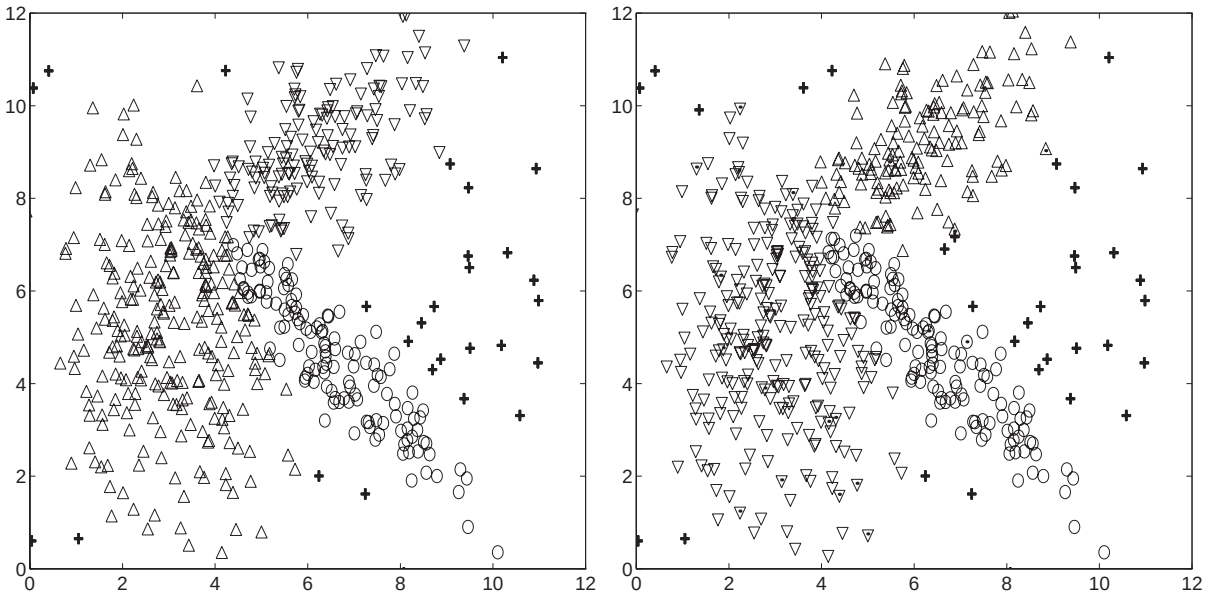


FIG. 4.9 – *Dif* : Partitionnement obtenu par les modèles #4 (à gauche) et notre modèle (à droite)- les + sont les points rejetés

modèles #3, #4 et #5) est faible (< 4) pour éviter de prendre en compte trop de données. La figure (4.9) montre les résultats avec les modèles #4 et #5. Pour le modèle #4, le rejet est effectué lorsque la probabilité a posteriori de la loi uniforme est majoritaire (critère MAP sur les 4 classes).

Nous avons également testé la robustesse de chacune des méthodes à l'initialisation aléatoire des centres des classes. Ici encore, nous avons sélectionné le paramétrage donnant l'erreur minimale en resubstitution sur les trois premières classes. La table (4.5) contient la moyenne et l'écart-type de l'erreur pour les 50 initialisations aléatoires différentes. Notre algorithme est de loin le meilleur avec une moyenne d'erreur proche de la valeur minimale et un écart-type faible. Ainsi, pour des valeurs raisonnables des paramètres, l'erreur moyenne obtenue est significative. cette robustesse peut s'expliquer par l'introduction du bi-mode qui élargit la queue de la distribution recherchée et permet d'éviter des minima locaux. Cette propriété est intéressante dans le contexte d'une application réelle.

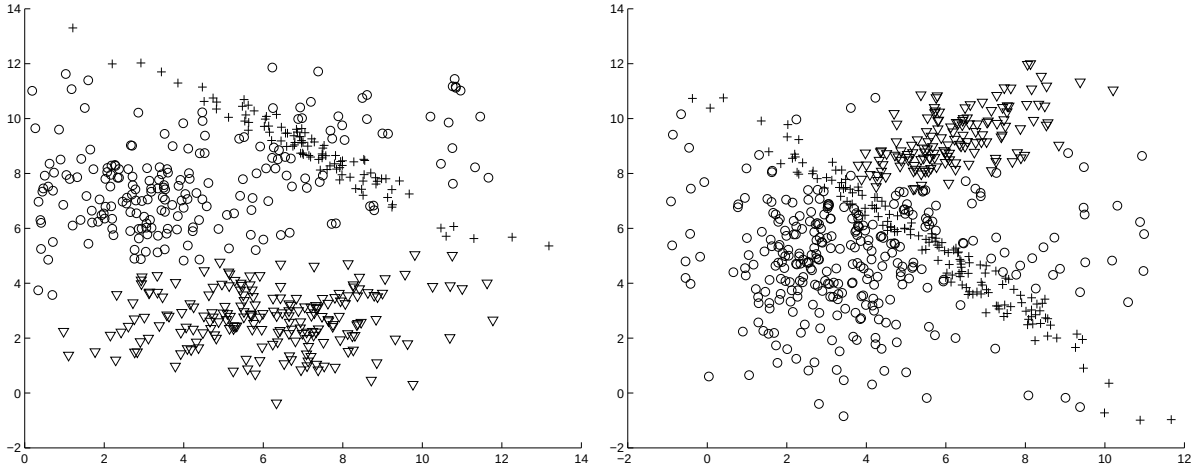


FIG. 4.10 – En l'absence d'estimation robuste, EM cherche à rendre vraisemblable les données bruitées

	Erreur	Modèle #1	Modèle #2	Modèle #3	Modèle #4	Modèle #5
Easy	Moyenne	8.31%	9.87%	9.86%	11.87%	2.96%
	Écart-type	10.87	12.57	15.18	15.15	0.36
Dif	Moyenne	18.82%	13.27%	16.76%	13.95%	11.93%
	Écart-type	5.8	4.27	5.	6.97	0.2

TAB. 4.5 – Résultats pour 50 initialisations aléatoires

4.2.5.2 Application à quelques jeux de données réels

Nous avons testé notre algorithme sur 4 jeux de données réels issus de la base de données UCI [BM98a]. Avant de pouvoir utiliser certains jeux de données, il est souvent nécessaire d'effectuer une transformation de celui-ci. En effet, chaque attribut possède généralement un domaine de définition propre (Ex. : Niveau de gris $[0; 100]$ et la teinte

$]-\pi;\pi]$). Ainsi, un déplacement unitaire n'aura pas de la même importance pour tous les attributs. L'une des techniques classiques pour résoudre ce problème est d'effectuer un centrage-réduction du jeux de données. Il consiste à appliquer une transformation linéaire de χ en χ' pour obtenir $E(\chi') = 0$ et $\sigma_m^2(\chi') = 1$ pour $m = 1, \ldots, p$. Pour un élément x_i de χ, on écrit

$$x_i'^m = \frac{x_i^m - \mu_m}{\sigma_m} \tag{4.14}$$

où μ_m et σ_m désignent respectivement la moyenne et l'écart-type déterminés empiriquement à partir de χ. Cette transformation, appelé centrage-réduction, n'affecte pas la corrélation entre les variables. Nous avons préféré utiliser la transformation de Mahalanobis qui, en plus, supprime la corrélation entre les variables :

$$x_i' = \Sigma^{\frac{-1}{2}}(x_i - \mu) \tag{4.15}$$

μ et Σ désignent le vecteur moyenne et matrice de covariance sur χ. La matrice de covariance de χ' se réduit à la matrice identité. Sur le plan pratique, cette transformation est très utile lors de l'inversion d'une matrice symétrique définie positive M très mal conditionnée (Ex. : Inversion de la matrice de covariance Σ_k pour la classe k). Une astuce algorithmique consiste à ajouter le terme $\epsilon\Sigma$ à M pour améliorer son conditionnement (ϵ faible). La nouvelle matrice à inverser devient $M + \epsilon\Sigma$. En utilisant la transformation de Mahalanobis, cette astuce se réduit à ajouter ϵ pour tous les termes non diagonaux.

Les 4 jeux de données utilisés sont *Breast-Cancer Wisconsin*, *Pima Indian diabetes*, *Iris*, *Wine*. Voici une brève description de chacun d'eux :

- *Breast-Cancer Wisconsin* : le problème considéré traite de la détection des cellules cancéreuses pour le diagnostic du cancer du sein. Les attributs extraits, au nombre de 10, proviennent d'une série d'observations annotée de 0 à 10 par un expert concernant la forme, la taille, l'opacité de la cellule. Ce problème à deux classes (maligne/begnine) est plutôt considéré comme facile (> 90 % de succès)
- *Pima Indian diabetes* : l'objectif est de diagnostiquer le diabète chez les femmes d'une tribu indienne aux États-Unis suivant des critères définis par l'OMS. Les attributs pris en compte sont de nature très diverses comme par exemple, le nombre de fois où la femme a été enceinte, la tolérance à l'absorption de glucose, la pression sanguine, l'indice de masse corporelle. Ce problème de classification binaire est connu pour être difficile car les attributs considérés ne sont pas suffisamment discriminants et les deux classes sont très mélangées.
- *Iris* : Ce jeu de données est sans doute l'un des plus connus et utilisés en reconnaissance des formes. L'objectif est de déterminer 3 variétés d'iris en utilisant les informations de longueur et largeur des pétales et des sépales. Une des classes est linéairement séparable des deux autres, alors que ces deux dernières ne sont pas linéairement séparables. Le problème du nombre de composantes pour ce jeu de données a déjà été abordé au chapitre 2.

– *Wine* : Ce problème considéré est la classification de vignes de vin italiennes issues de trois cultivateurs. Il s'agit de déterminer la provenance d'un vin à partir de 13 caractéristiques physico-chimiques. Ce jeu de données, tenu pour très facile, est souvent utilisé pour valider une nouvelle méthode de classification.

La table (4.6) compare les taux d'erreur obtenus par notre méthode avec ceux obtenus les algorithmes CEM, EM, SEM, FCM. Les trois valeurs données pour notre méthode correspondent aux taux de succès, d'erreur et de rejet associés au paramétrage sélectionné par le critère $C(\Gamma)$ (voir Eq. (4.8)). A titre d'indication, nous avons également reporté le taux d'erreur de l'analyse factorielle discriminante. Les implémentations des algorithmes CEM, EM et SEM sont celles du logiciel XEMgaus sous Matlab [Bie99]. La méthode FCM a été également implémentée en Matlab. Il est à noter que pour le jeu *Breast-Cancer Wisconsin*, nous avons été obligé de contraindre à l'égalité les volumes des deux matrices de covariance pour CEM, EM et SEM (cf. décomposition de la matrice de covariance). Sans cela, toutes les initialisations, même aléatoires, conduisent à inverser des matrices singulières induisant un échec de ces algorithmes.

	CEM	EM	SEM	FCM	Notre méthode	AFD
Breast-Cancer Wisconsin n=683,c=2,p=9	15.23%	14.79%	14.79%	4.39%	S=91.95%, E=5.12%, R=2.93%	3.66%
Pima Indian diabetes n=768,c=2,p=8	35.42%	35.94%	35%	34.11%	S=66.93%, E=29.04%, R=4.04%	23.18%
Iris n=150,c=3,p=4	4%	3.33%	3.33%	10.67%	S=96.67%, E=2.66%, R=0.67%	12.67%
Wine n=178,c=3,p=13	11.24%	6.18%	5.62%	25.28%	S=89.32%, E=5.62%, R=5.06%	100%

TAB. 4.6 – Comparaison du taux d'erreur sur divers jeux de données réels

Concernant le jeu de données *Breast-Cancer Wisconsin*, la meilleure méthode non supervisée se révèle être l'algorithme FCM. Tous les attributs possédant les mêmes domaines de définition ([0 ;10]), la métrique euclidienne non pondérée utilisée dans FCM est suffisante. Notre algorithme est un peu moins performant que FCM mais surclasse EM et ses variantes. Pour *Pima Indian diabetes* et *Iris*, nous obtenons le meilleur résultat avec un taux de succès équivalent mais un taux d'erreur plus faible ($\sim -6\%$ et $\sim -0.67\%$).

L'échec relatif de FCM sur *Iris* et *Wine* s'explique par l'utilisation de la distance euclidienne non pondérée qui conduit à rechercher des classes sphériques. Sur ce dernier jeu de données, il est à noter que le même algorithme commet 31,46% d'erreur lorsque les données sont non normalisées. Pour *Wine* également, notre méthode obtient un taux comparable à EM et à SEM, mais le rejet de points valides fait diminuer le taux de succès.

Nous avons testé les résultats de notre algorithme avec différentes configurations des paramètres de coûts dans $C(\Gamma)$ et on constate que

- Majorer C_c, le coût associé au taux de succès, conduit à ne pas rejeter pour éviter de rejeter des points valides.
- Majorer C_e, le coût associé au taux d'erreur, conduit à rejeter beaucoup de points et produit des classes plus concentrées sur les zones de fortes densités (autour de leur noyau).
- Majorer C_r revient à majorer C_c, mais on doit imposer $C_c > C_e$ car maximiser

$$C(\Gamma) \simeq C_c P_c - C_e(1 - P_c) = (C_c - C_e)P_c + C_e \tag{4.16}$$

  doit conduire à maximiser P_c.

4.3 Extension à la semi-supervision

La méthode proposée peut être étendue de manière naturelle au contexte de la supervision partielle [SJF01b]. Dans le chapitre 3, nous avons présenté l'approche probabiliste de la semi-supervision pour la recherche de classes multi-modales. Nous nous limitons ici à une présentation du cas de classes uni-modales en soulignant les éventuels problèmes d'une extension aux classes multi-modales.

4.3.1 Principe

Dans le cadre probabiliste et pour un algorithme comme EM, la semi-supervision s'effectue naturellement en fixant les probabilités a posteriori des points supervisés. Comme dans la version ssFCM de Bensaïd pour les degrés d'appartenance, on ne réestime pas ces probabilités a posteriori lors de l'étape d'estimation de EM. Dans le cas d'une supervision binaire, le vecteur d'appartenance est

$$z_{ik} = P(C_k|x_i) = \begin{cases} 1 & \text{si } x_i \text{ appartient à la classe } C_k \\ 0 & \text{sinon} \end{cases} \tag{4.17}$$

On peut s'interroger si l'on doit utiliser les points supervisés pour l'estimation initiale des paramètres des classes. Cela revient à poser la question de leur représentativité vis-à-vis des classes, i.e. "est-ce que les points supervisés sont distribués selon la même loi

régissant l'ensemble des données ?". Le problème se pose lorsqu'il y a un biais de sélection ou que l'on dispose d'un nombre limité de points supervisés. Pour les probabilités a priori des classes par exemple, on peut utiliser la formule :

$$\hat{\pi}_k = \frac{\sum_{x_i \in \chi^d} z_{ik}}{n_d} \tag{4.18}$$

où n_d désigne respectivement le nombre de points supervisés et z_{ik} la probabilité a posteriori fixée par l'expert. Si la sélection est biaisée, cette estimation a peu de sens. Il convient donc de sélectionner rigoureusement les points supervisés.

Notre algorithme se décompose à la manière d'EM en 2 étapes. Grâce à la valeur initiale des paramètres des classes, il est possible de calculer la première valeur de la vraisemblance complète (Eq. : 2.10) pour établir le critère de convergence pour la première boucle globale. Dans une première étape, on estime les probabilités a posteriori des points non supervisés de manière classique :

$$\text{pour } x_i \in \chi^u,\ \hat{z}_{ik}^{(t)} = \frac{\hat{\pi}_k^{(t)} f(x_i; \hat{\Theta}_k^{(t)})}{\sum_{l=1}^{c} \hat{\pi}_l^{(t)} f(x_i; \hat{\Theta}_l^{(t)})} \tag{4.19}$$

Dans un second temps, on effectue la mise à jour des paramètres des classes. Si l'on fait l'hypothèse que la sélection des points supervisés est non biaisée, le calcul des probabilités a priori s'effectue par la formule :

$$\hat{\pi}_k^{(t+1)} = \frac{\sum_{x_i \in \chi^u} \hat{z}_{ik}^{(t)} + n_k}{n_u + n_d} \tag{4.20}$$

en utilisant l'estimation courante $\hat{z}_{ik}^{(t)}$ des probabilités a posteriori et où n_k désigne le nombre de points supervisés provenant de la classe k. Afin d'estimer les paramètres $\hat{\mu}_k, \hat{\Sigma}_k, \tilde{\mu}_k, \tilde{\Sigma}_k$, nous utilisons notre procédure robuste itérative. Les paramètres robustes sont initialisées à partir estimateurs classiques, ce qui donne pour la moyenne robuste :

$$\tilde{\mu}_k^{(t+1,0)} = \hat{\mu}_k^{(t+1,0)} = \frac{\sum_{x_i \in \chi^u} \hat{z}_{ik}^{(t)} x_i + \sum_{x_i \in \chi^d} z_{ik} x_i}{\sum_{x_i \in \chi^u} \hat{z}_{ik}^{(t)} + n_k} \tag{4.21}$$

Le premier terme du numérateur constitue la contribution des données non supervisées alors que le second donne celle des points supervisés. On remarquera que pour ce dernier terme, il n'est pas nécessaire d'utiliser la notation $\hat{z}_{ik}^{(t)}$ car ces probabilités sont constantes tout au long de l'algorithme. Une formulation analogue peut être formulé pour la matrice de covariance (voir algorithme 22). On effectue par la suite la procédure itérative d'estimation des paramètres robustes en sommant à chaque fois, la partie supervisée et non supervisée. Comme précédemment, il faut veiller à ne pas supprimer trop de points dans l'estimation en fixant le critère de convergence. Une fois que l'estimation robuste

des paramètres est effectuée, on doit évaluer la convergence de l'algorithme. Nous nous servons à la fois d'un critère sur la vraisemblance (t^* itérations sans améliorations de la vraisemblance complète) et d'une borne t_{max} du nombre d'itérations de l'algorithme. L'algorithme (22) décrit l'ensemble de notre algorithme en incluant l'estimation robuste des paramètres des classes.

4.3.2 Coopération Robustesse/Semi-supervision

Dans notre approche, il se pose le problème du choix du seuil de l'estimateur utilisé pour l'estimation robuste. Cela revient indirectement à estimer la dispersion d'une classe pour déterminer la taille du filtrage à appliquer et conserver un pourcentage des données. On peut utiliser un certain nombre d'estimateurs plus ou moins robustes de l'écart-type d'une distribution, dont les suivants classés par robustesse croissante :

- L'écart-type empirique :

$$S_n = \sqrt{\frac{1}{n-1}\sum(X_i - \bar{X})^2} \tag{4.22}$$

- L'écart absolu médian (MAD)

$$MAD_n = \frac{1}{\Phi^{-1}(\frac{3}{4})} * Med_i(|X_i - Med_j(X_j)|) \tag{4.23}$$

  où Φ^{-1} est la fonction réciproque de la fonction de répartition d'une variable normale centrée-réduite. $\Phi^{-1}(\frac{3}{4}) = 1.4826$

- L'étendue inter-quartile ("Interquartile range")

$$IQR_n = \frac{1}{2\Phi^{-1}(\frac{3}{4})}(X_{(3n/4)} - X_{(1n/4)}) \tag{4.24}$$

  où $X_{(j)}$ désigne le j-ème quantile.

Dans notre contexte d'étude, on dispose d'un terme de pondération constitué par le produit $w_i z_{ik}$. Si l'on note plus simplement ce produit w_i, il est alors possible utiliser l'écart-type pondéré :

$$\tilde{sd} = \sqrt{\frac{\sum_{i=1}^{n} w_i(x_i - \bar{x_w})^2}{\frac{(n^+-1)\sum_{i=1}^{n} w_i}{n^+}}} \tag{4.25}$$

où $\bar{x_w}$ désigne la moyenne empirique pondérée et n^+ le nombre de poids non nuls.

Dans le contexte semi-supervisé, nous proposons d'utiliser les points supervisés pour déterminer la taille d'une classe [SJF01a]. Sachant que l'on ne remet pas en cause ces

Algorithme 22: Un algorithme de classification robuste semi-supervisée

Données : $\chi^u = \{x_1, \ldots, x_{n_u}\}$ un ensemble de données non supervisées, $\chi^d = \{(x_1, z_1), \ldots, (x_{n_d}, z_{n_d})\}$ un ensemble de données supervisées, c le nombre de classes recherchées, ψ la fonction d'influence du M-estimateur, γ et α les paramètres du modèle des classes, $\theta^{(0)}$ la valeur initiale des paramètres du mélange, t^* nombre maximal d'itérations sans amélioration de la vraisemblance, t_{max} nombre maximal d'itérations.

Sortie : π_k, $\hat{\mu}_k$, $\hat{\Sigma}_k$, $\tilde{\mu}_k$, $\tilde{\Sigma}_k$

$t \leftarrow 0$;

Calculer la vraisemblance $L^{(0)}$ de $\theta^{(0)}$;

pour *k = 1 à c* **faire** $n_k = \sum_{x_i \in \chi^d} z_{ik}$;

répéter

Calcul des probabilités a posteriori des points non supervisés $\hat{z}_{ik}^{(t)}$:

$$\text{pour } x_i \in \chi^u,\ \hat{z}_{ik}^{(t)} = \frac{\hat{\pi}_k^{(t)} f(x_i; \hat{\Theta}_k^{(t)})}{\sum_{l=1}^{c} \hat{\pi}_l^{(t)} f(x_i; \hat{\Theta}_l^{(t)})}$$

Mise à jour des paramètres du mélange

$$\hat{\pi}_k^{(t+1)} = \frac{\sum_{x_i \in \chi^u} \hat{z}_{ik}^{(t)} + n_k}{n_u + n_d}$$

$\tau \leftarrow 0$;

$\tilde{\mu}_k^{(t+1,0)} = \hat{\mu}_k^{(t+1,0)} = \frac{\sum_{x_i \in \chi^u} \hat{z}_{ik}^{(t)} x_i + \sum_{x_i \in \chi^d} z_{ik} x_i}{\sum_{x_i \in \chi^u} \hat{z}_{ik}^{(t)} + n_k}$;

$\tilde{\Sigma}_k^{(t+1,0)} = \hat{\Sigma}_k^{(t+1,0)} = \frac{\sum_{x_i \in \chi^u} \hat{z}_{ik}^{(t)} (x_i - \hat{\mu}_k^{(t+1,0)})(x_i - \hat{\mu}_k^{(t+1,0)})^T + \sum_{x_i \in \chi^d} z_{ik} (x_i - \hat{\mu}_k^{(t+1,0)})(x_i - \hat{\mu}_k^{(t+1,0)})^T}{\sum_{x_i \in \chi^u} \hat{z}_{ik}^{(t)} + n_k}$;

répéter

pour $x_i \in \chi^u \bigcup \chi^d$ **faire**

$e_i^{(\tau-1)} = d_{\mathcal{M}}(x_i; \tilde{\mu}_k^{(t+1,\tau-1)}, \tilde{\Sigma}_k^{(t+1,\tau-1)})$;

$w_i^{(\tau-1)} = \frac{\psi(e_i^{(\tau-1)})}{e_i^{(\tau-1)}}$;

$\tilde{\mu}_k^{(t+1,\tau)} = \frac{\sum_{x_i \in \chi^u} w_i^{(\tau-1)} \hat{z}_{ik}^{(t)} x_i + \sum_{x_i \in \chi^d} w_i^{(\tau-1)} z_{ik} x_i}{\sum_{x_i \in \chi^u} w_i^{(\tau-1)} \hat{z}_{ik}^{(t)} + \sum_{x_i \in \chi^d} w_i^{(\tau-1)} z_{ik}}$;

$\tilde{\Sigma}_k^{(t+1,\tau)} = \frac{\sum_{x_i \in \chi^u} w_i^{(\tau-1)} \hat{z}_{ik}^{(t)} (x_i - \tilde{\mu}_k^{(t+1,\tau)})(x_i - \tilde{\mu}_k^{(t+1,\tau)})^T + \sum_{x_i \in \chi^d} w_i^{(\tau-1)} z_{ik} (x_i - \tilde{\mu}_k^{(t+1,\tau)})(x_i - \tilde{\mu}_k^{(t+1,\tau)})^T}{\sum_{x_i \in \chi^u} w_i^{(\tau-1)} \hat{z}_{ik}^{(t)} + \sum_{x_i \in \chi^d} w_i^{(\tau-1)} z_{ik}}$;

$\tau \leftarrow \tau + 1$;

jusqu'à *Critère d'arrêt ou de convergence*;

Calculer la vraisemblance $L^{(t+1)}$ de $\theta^{(t+1)}$;

jusqu'à $t < t_{max}$ *ou* $L^{(t)}$ *non améliorée depuis* t^* *itérations*;

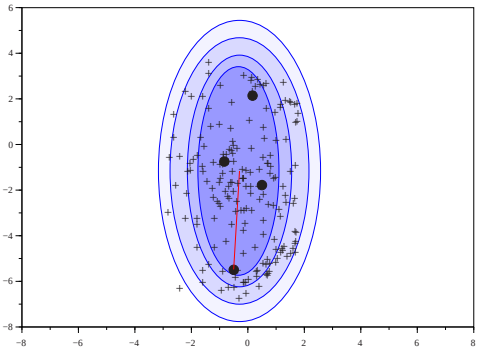


FIG. 4.11 – Détermination du seuil du M-estimateur de Huber à l'aide des points supervisés (•)

points, on considère que ceux-ci ainsi que les points non supervisés plus proches du prototype de la classe (au sens de la distance de Mahalanobis) sont significatifs pour l'estimation. Cette heuristique peut s'écrire dans le cadre de notre estimation robuste itérative comme suit :

$$h_k = \max_{x_i \in C_k} (d_{\mathcal{M}}(x_i, C_k), h_{min}) \tag{4.26}$$

Il est indispensable de fixer un minimum h_{min} à h sans lequel l'estimation peut porter que sur un trop petit nombre de points. En effet, cette situation se produit si l'on dispose d'un nombre très limité de points supervisés, par exemple 1, proche de la moyenne estimée. On peut imaginer fixer le seuil h_{min} par l'écart-type pondéré (voir Eq. 4.25). On peut remarquer que ce critère nécessite une supervision binaire des points.

Si l'on choisit d'utiliser le M-estimateur de Huber (voir table 3.1), le paramètre h associé à cet estimateur correspond à la taille d'une fenêtre pour laquelle le poids est maximal (voir Fig 3.1). La figure 4.11 représente les divers paliers du M-estimateur de Huber issus de cette heuristique, du plus foncé (poids le plus fort) au plus clair (poids le plus faible). Cette heuristique ne semble pas être applicable telle quelle au cas de classes multi-modales car on est conduit à considérer les mêmes points supervisés sans différenciation des sous-composantes.

Cette estimation du seuil du M-estimateur s'insère dans notre algorithme avant le calcul des poids associés à chacun des points. Il est ainsi recalculé pour chaque itération de l'estimation robuste itérative. Dans ce contexte, le coût algorithmique de cette procédure est primordial. Notre méthode est très peu coûteuse en temps (et en espace) car il suffit d'un parcours des points supervisés ($\Theta(n_d)$). A l'inverse, un estimateur comme IQR nécessite la sélection de deux quantiles (1/4 et 3/4) qui peut être effectuée en $\mathcal{O}(nlog(n))$

si on utilise un tri.

4.3.3 Expérimentations

Nous avons testé notre algorithme sur des données artificielles et réelles précédemment utilisées pour pouvoir quantifier l'apport de la supervision. La sélection des points supervisés a été effectuée par tirage aléatoire sans tenir compte des proportions des classes. Si le tirage n'est pas favorable, l'amélioration légitimement attendue peut être moindre. D'autre part, le nombre de paramètres de l'algorithme est trop important pour explorer complètement des intervalles de ceux-ci sur tous les jeux de données. Ainsi, nous avons choisi de faire un tirage aléatoire sur les γ, α et taux de supervision pour diverses initialisations. Le M-estimateur choisi est celui de Huber dont nous avons sélectionné le paramètre avec la méthode décrite précédemment qui utilise les points supervisés. Nous comparons notre méthode à celle de Pedrycz sur les jeux de données *Easy* et *Dif* et *Iris*. Pour cet algorithme, nous avons cherché des paramètres optimaux. Pour *Iris*, nous reportons en plus les résultats de la méthode de Bensaïd publiés dans [LBB98]. Pour le calcul du taux d'erreur, nous effectuons une resubstitution sans tenir de la classe des points supervisés. En effet, tenir compte de cet étiquetage fausse l'estimation de l'erreur, car superviser 100% des points conduirait alors à estimer systématiquement le taux de succès à 100%.

Sur le jeu de données *Easy* (classes séparées), nous avons amélioré le taux d'erreur sans supervision (2.86%) à 0.28% pour un taux de rejet de 10% et un taux de supervision de 9%. Les points rejetés sont exclusivement issus de la distribution uniforme. Ainsi, aucun point des trois classes gaussiennes n'a été rejeté à tort. En supervisant les mêmes points, la méthode de Pedrycz obtient 1.14% sans toutefois effectuer du rejet. La figure (4.12) montre le résultat des deux algorithmes.

Sur les données *Dif* (classes mélangées), nous avons également diminué le taux d'erreur (8.44%) avec 8.67% de supervision par rapport au meilleur taux obtenu précédemment sans supervision (11.33%) tout en rejetant à tort 2 points. Sur ce jeu de données, la méthode de Pedrycz est meilleure que la nôtre avec 6.6% d'erreur.

Concernant les 4 jeux de données réels, les figures (4.14), (4.16), (4.18), (4.20) montrent respectivement les meilleurs résultats obtenus au sens du critère $C(\Gamma)$ parmi les diverses configurations paramétriques aléatoires ($\gamma \in [0.05, 0.5]$, $\alpha \in [1, 20]$ et le numéro d'initialisation de 1 à 50) pour différents taux de supervision avec des paramètres de coût fixés comme suit : C_c=1, C_e=2, C_r =2. Sur chaque figure, l'abscisse représente le taux de supervision (de 0 à 100%), l'axe de gauche, le taux de succès, l'axe de droite le taux d'erreur. Dans les figures (4.15), (4.17), (4.19), (4.21), le taux d'erreur est remplacé par le taux de rejet. L'aspect chaotique des courbes s'explique par le choix aléatoire des paramètres.

Comme on pouvait s'y attendre, l'allure générale de ces courbes confirme que lorsque le

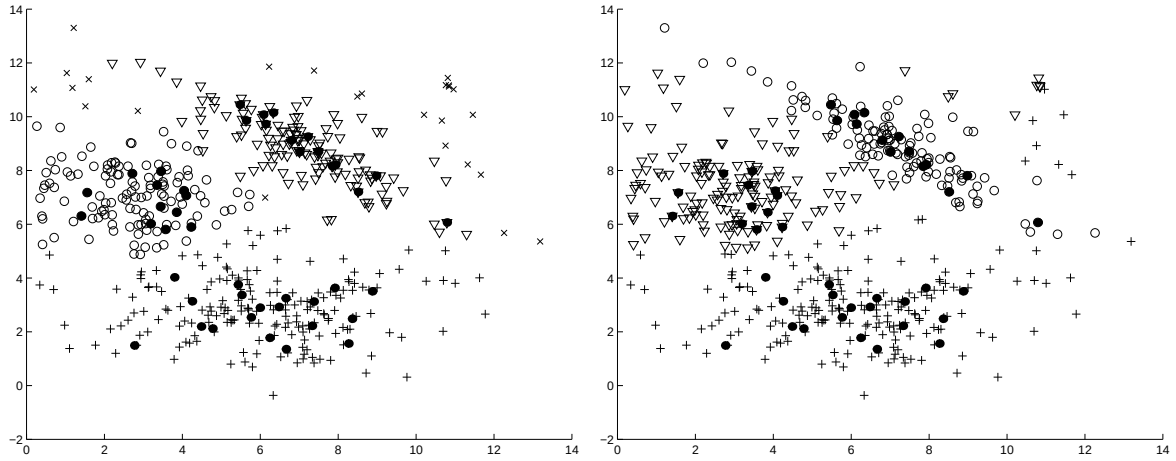


FIG. 4.12 – *Easy* : Résultats de notre algorithme (à gauche) et Pedrycz (à droite)

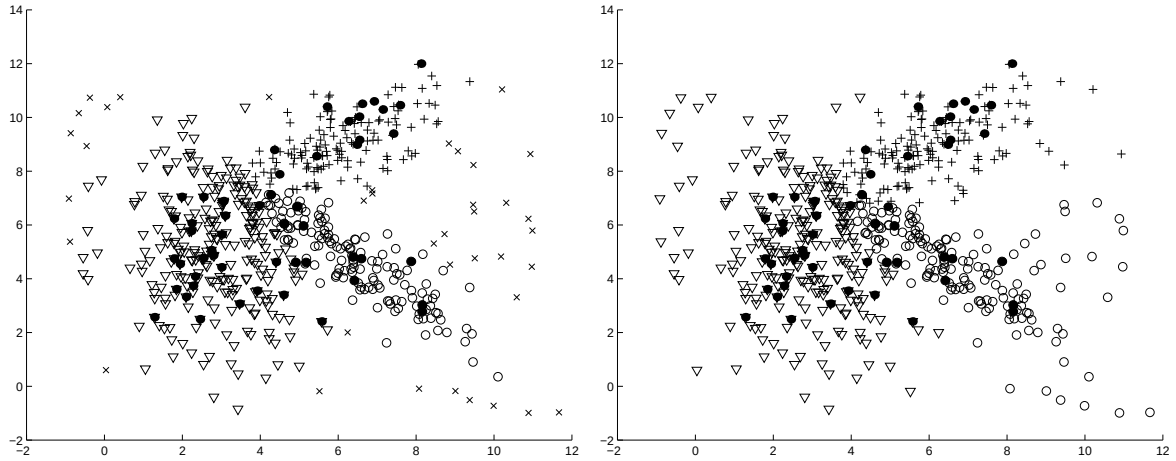


FIG. 4.13 – *Dif* : Résultats de notre algorithme (à gauche) et Pedrycz (à droite)

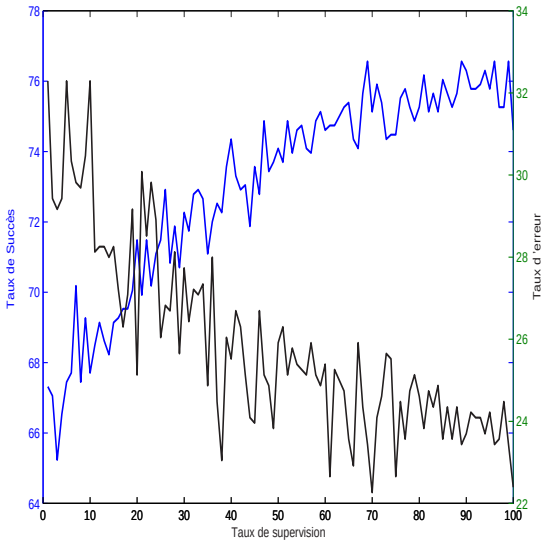


FIG. 4.14 – *Pima Indian Diabetes* : Taux de succès et d'erreur en fonction du taux de supervision

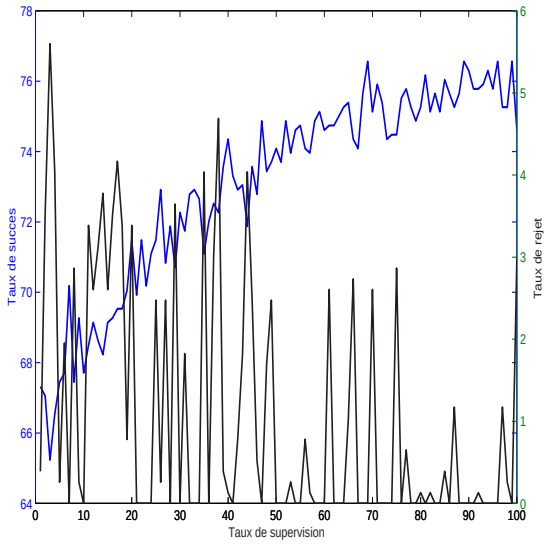


FIG. 4.15 – *Pima Indian Diabetes* : Taux de succès et de rejet en fonction du taux de supervision

taux de supervision augmente, le succès augmente et l'erreur diminue. On constate aussi que l'amélioration est plus rapide pour les jeux de données *Iris* et *Wine* (plus faciles). Quelques résultats extraits de ces courbes sont reportés dans la table (4.7).

Pour *Iris*, nous avons obtenu un taux d'erreur de 1.33% pour 5.33% de supervision. En utilisant les mêmes points supervisés, la méthode de Pedrycz commet 10% d'erreur. A titre de comparaison, Labzour et al. ont obtenu 2% d'erreur en supervisant 5 points par classe, soit un taux de 10% de supervision [LBB98]. Concernant le jeu de données *Wine*, on retrouve les mêmes résultats que l'AFD (100%), en effectuant une supervision totale.

Pour les données *Pima Indian Diabetes*, nous avons obtenu un taux d'erreur (16.66%) pour un taux de rejet important (15.74%) grâce à une pénalisation forte de l'erreur ($C_e \gg 2$). En contrepartie, le taux de succès a été abaissé à 16.66%. Sur ce jeu de données, les meilleures méthodes supervisées sans rejet commettent entre 20% et 22% d'erreur.

De manière générale, on constate que superviser au delà d'un certain seuil ne permet pas d'améliorer le taux d'erreur. En revanche, on observe de manière évidente une amélioration continuelle de l'estimation des paramètres des classes qui tendent vers les paramètres théoriques.

L'abaque de la figure 4.22 montre, pour divers niveaux de supervision, la relation entre le taux de rejet et le taux d'erreur. Comme on pouvait s'y attendre, le rejet compense l'erreur potentielle de classification qui diminue lorsque la supervision augmente. De plus,

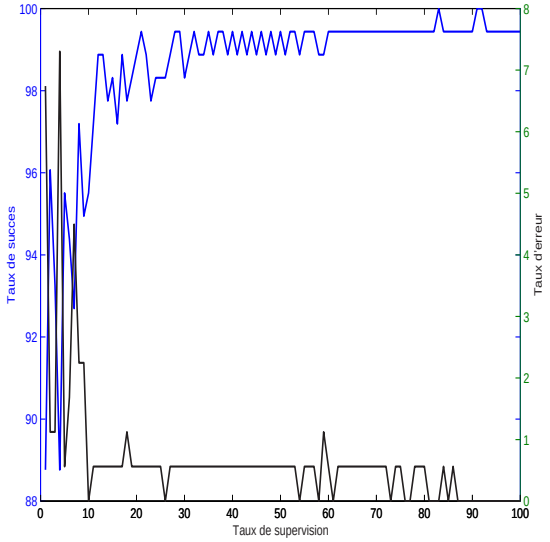


FIG. 4.16 – *Wine* : Taux de succès et d'erreur en fonction du taux de supervision

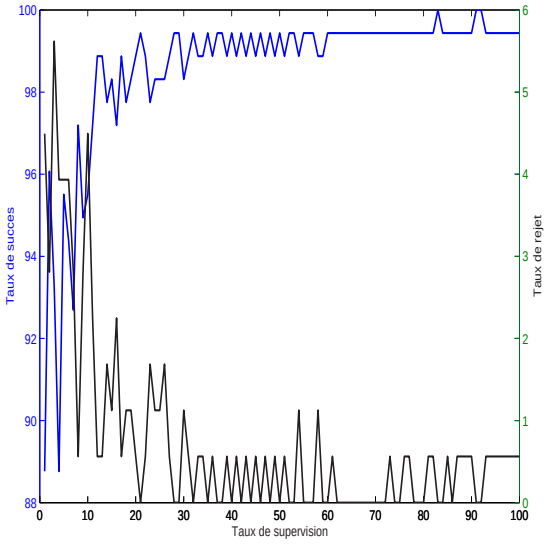


FIG. 4.17 – *Wine* : Taux de succès et de rejet en fonction du taux de supervision

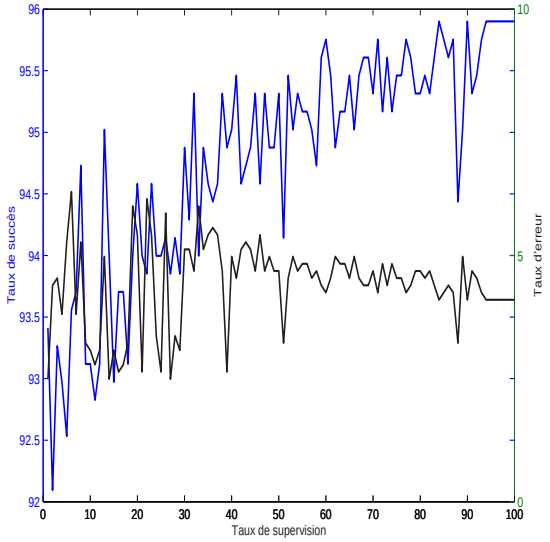


FIG. 4.18 – *Breast Cancer Wisconsin* : Taux de succès et d'erreur en fonction du taux de supervision

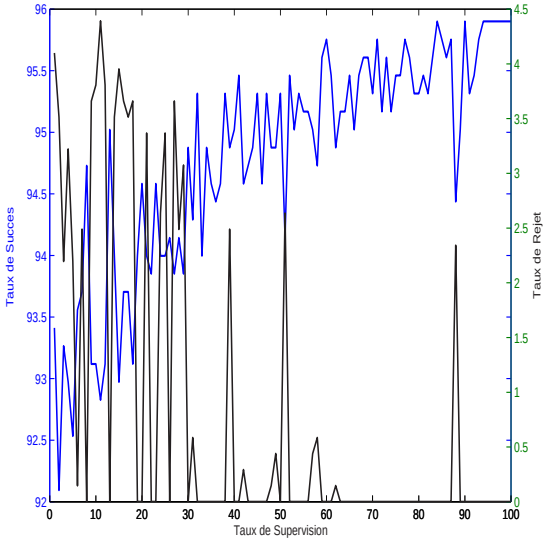


FIG. 4.19 – *Breast Cancer Wisconsin* : Taux de succès et de rejet en fonction du taux de supervision

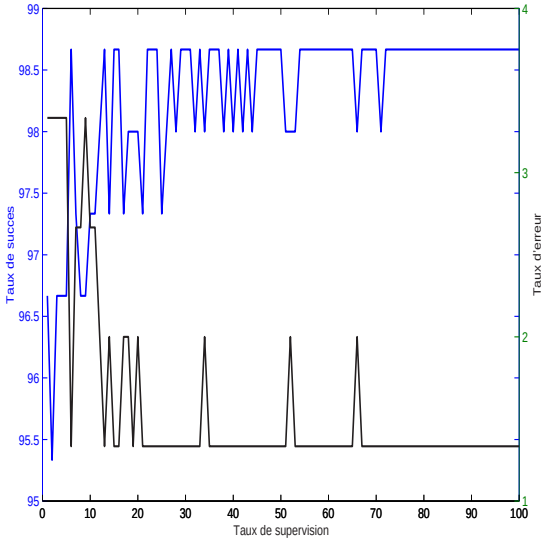


FIG. 4.20 – *Iris* : Taux de succès et d'erreur en fonction du taux de supervision

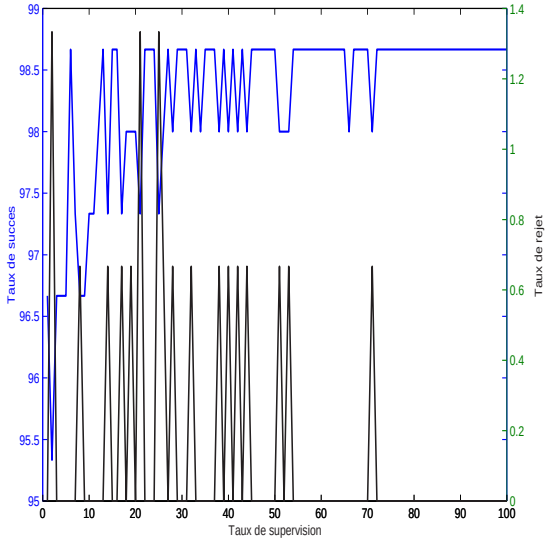


FIG. 4.21 – *Iris* : Taux de succès et de rejet en fonction du taux de supervision

Jeu de données	Taux de supervision	Taux de succès	Taux d'erreur	Taux de Rejet
Breast Cancer Wisconsin	0.%	91.95%	5.12%	2.93%
	1.02%	93.41%	2.49%	4.1%
	100.%	95.9%	4.1%	0.%
Iris	0.%	96.67%	2.66%	0.67%
	5.33%	98.66%	1.33%	0.%
	100.%	98.66%	1.33%	0.%
Pima Indians Diabetes	0.%	66.93%	29.04%	4.04%
	6.77%	70.18%	29.82%	0.%
	100.%	76.56%	23.44%	0.%
Wine	0.%	89.32%	5.62%	5.06%
	1.78%	96.07%	1.12%	2.81%
	100.%	100.%	0.%	0.%

TAB. 4.7 – Résultats extraits pour les jeux de données réels

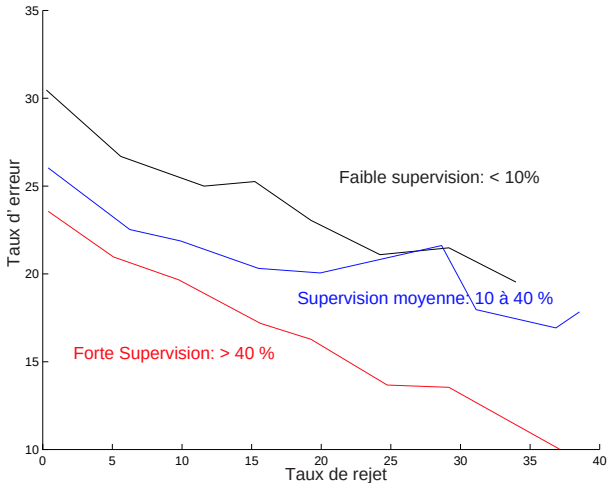


FIG. 4.22 – Abaque du taux d'erreur avec le taux de rejet pour divers niveaux de supervision pour le jeu de données *Pima Indian Diabetes*

nous avons constaté que le nombre d'itérations effectuées par notre algorithme est plus faible que celui de l'algorithme EM même si chaque itération dure plus longtemps du fait de l'estimation itérative robuste. Cela provient du fait que notre estimation itérative se concentre plus précisément sur l'estimation des paramètres d'une classe à la fois (comme "Component-Wise EM"). En effet, on peut voir l'estimation itérative comme une version locale de l'algorithme EM à une classe faisant l'analogie entre le calcul des pondérations et l'estimation des probabilités a posteriori et entre la réestimation des paramètres de la classe et l'étape de maximisation de EM. Ainsi, le surcoût algorithmique de l'estimation itérative est compensé en partie par la diminution du nombre d'itérations de la boucle globale.

Conclusion Générale

La reconnaissance des formes est un domaine de recherche qui, aujourd'hui, donne lieu à de nombreuses études et à des approches très diverses. Dans cette problématique, nous nous sommes plus particulièrement intéressés à l'approche statistique et aux méthodes paramétriques de la classification automatique. Le premier constat ayant motivé cette étude est l'échec de la plupart des méthodes de classification en présence de données aberrantes. La justification de ce phénomène réside dans le fait que l'influence des données aberrantes n'est pas bornée. Au cours de l'estimation des paramètres des classes, ce type de données biaise les estimateurs non robustes, ce qui rend certains algorithmes de classification impropres à l'utilisation en leur présence. Par exemple, nous avons vu que l'algorithme des C-moyennes floues est mis en défaut par la recherche d'un maximum de séparabilité entre les classes. De même et pour un algorithme comme EM, le principe du maximum de la vraisemblance conduit à rechercher un modèle paramétrique rendant plausible les données aberrantes.

Le première partie de cette thèse a concerné l'étude d'approches robustes de la classification. Pour borner l'influence des points aberrants, nous avons distingué et présenté trois méthodes possibles :

- L'utilisation d'estimateurs robustes des paramètres des classes qui utilisent des fonctions d'influence (M-estimateurs),
- La création d'une classe supplémentaire de bruit,
- L'inclusion d'un modèle de contamination dans le modèle théorique de la classe.

Nous avons proposé une méthode originale de classification automatique basée sur une modélisation théorique particulière des classes. L'idée fondamentale de cette méthode est d'estimer de manière différenciée (robuste et classique) les mêmes variables aléatoires pour former les deux sous-composantes du mélange modélisant une classe. Les paramètres de la première sous-composante, associée aux données "propres" de la classe (noyau), sont estimés par une procédure itérative robuste. Cette dernière est fondée sur une repondération des erreurs à chaque itération permettant de limiter l'influence des points aberrants. A l'opposé, nous utilisons une estimation classique non robuste des paramètres du second mode. En présence de données aberrantes, seuls les paramètres de seconde sous-composante seront perturbés. Ainsi, notre méthode combine à la fois des estimateurs

robustes et l'utilisation d'un modèle de contamination des données. Des expérimentations menées sur des données artificielles ont permis de montrer que même lorsque les données bruitées sont distribuées différemment du modèle de contamination, l'estimation des paramètres des classes est très satisfaisante. Le choix des divers paramètres du modèle reste un problème ouvert, dont la solution est non traviale et pourtant au combien utile pour les praticiens. C'est une des pistes qu'il convient maintenant d'explorer pour améliorer la méthode proposée. On peut fort bien imaginer une procédure auto-adaptative permettant d'estimer conjointement le paramètre de proportion entre les deux sous-composantes par classe et le paramètre d'étalement. Nous allons nous pencher sur ce sujet.

Nous avons ensuite proposé une interprétation du modèle théorique d'une classe permettant d'introduire le rejet en distance. Celui-ci est fondé sur la création d'une classe supplémentaire regroupant l'ensemble des données contaminées de chacune des classes. Ainsi, alors que les deux sous-composantes formant le modèle d'une classe sont estimées conjointement, celles-ci sont séparées lors de la phase d'affectation. Les expérimentations que nous avons menées sur divers jeux de données réels et artificiels ont montré que nous améliorons, dans la majeure partie des cas, les résultats obtenus par EM et ses variantes, et par FCM, deux algorithmes couramment utilisés en classification. Le taux de rejet est contrôlé indirectement par le choix des paramètres du modèle. Dans le cas où l'on dispose d'un étiquetage des données, nous proposons de choisir a posteriori le modèle adéquat par une règle effectuant un compromis entre l'erreur et le rejet. Nous avons observé une robustesse certaine aussi bien vis-à-vis des données aberrantes qu'à l'initialisation des paramètres.

La seconde partie de ce travail a concerné l'étude des méthodes de classification semi-supervisée. La semi-supervision peut se définir de manière duale comme l'utilisation des données étiquetées pour un algorithme de classification automatique ou l'utilisation d'un ensemble de données non supervisées pour l'amélioration d'une méthode supervisée. Nous avons ainsi dressé un état de l'art des algorithmes de classification semi-supervisée. Nous avons vu que la supervision partielle revient essentiellement à contraindre le processus d'estimation en accordant une importance supérieure aux données étiquetées. Cela peut être effectué en modifiant la fonctionnelle réalisant la classification (Ex. : FCM) ou par un critère local (Ex. : Machines à Vecteurs de Support). Nous avons étendu notre méthode de classification robuste à la semi-supervision. Il s'agit en fait de fixer les probabilités a posteriori des points supervisés, que l'on ne réestime plus, en fonction de l'expertise. Nous avons également proposé une méthode de sélection du seuil d'un M-estimateur utilisant des données supervisées. Diverses expériences ont montré une substantielle amélioration dès qu'un faible nombre de points étiquetés sont disponibles. L'apport de la semi-supervision est supérieur pour des problèmes plus difficiles où les classes se chevauchent. A l'inverse, le bénéfice est moindre lorsque les classes sont séparées.

Nous avons observé qu'une supervision partielle non adaptée au couple (Données, Algorithme) induit un effet inverse de celui recherché. Cela peut être le cas lorsque des points supervisés imposent une contrainte géométrique non conforme au modèle recherché par un algorithme. Nous pensons qu'il faut se donner les moyens de maximiser l'apport de l'expertise, soit en sélectionnant de manière automatique une partie seulement des points supervisés, soit en proposant à un éventuel expert de superviser des points d'intérêt. La stratégie de sélection des points supervisés est dans les deux cas un problème lui aussi ouvert, le second apportant une dimension interactive au procédé de semi-supervision. Les questions auxquelles il convient de répondre sont : faut-il mieux superviser des points dans les zones de recouvrement des classes, proches des noyaux, ou alors dans les zones extérieures moins denses ? C'est également sur cet aspect de la problématique de la classification en mode partiellement supervisé que vont porter nos recherches futures.

Enfin, nous souhaitons vivement valider la méthode par une application réelle. Nous avons d'ores et déjà entrepris de le faire dans le cadre du projet Aqu@thèque pour lequel il s'agit de reconnaître des poissons évoluants dans un aquarium.

Annexe A

Interprétation de la matrice de confusion

Lorsqu'on effectue une classification automatique des données, on est confronté au problème de la validation des classes résultantes par rapport aux classes initiales. Ce problème connu également sous le nom de "label switching" correspond à l'existence d'une permutation σ entre les deux systèmes de numérotation. Dans [Ste00], Stephens aborde le cas général de l'identification des composantes sans se baser sur des connaissances a priori sur les classes (Ex. : paramètres théoriques). On se restreindra ici au cas où l'on connaît l'étiquetage réel des individus. Pour détecter d'éventuelles permutations, il est alors possible de se servir de la matrice de confusion.

La matrice de confusion est obtenue en comparant les données classées (en colonnes) avec des données de référence (en lignes). L'élément $C_{i,j}$ désigne le nombre de points de la classe originelle C_i classée par l'algorithme dans la classe Π_j. Voici un exemple de matrice de confusion :

	Π_1	Π_2	Π_3	Π_4	Total
C_1	93	1	3	4	101
C_2	5	75	23	4	107
C_3	2	32	86	1	121
C_4	0	1	5	141	147
Total	100	109	117	150	476

Dans le cas idéal (sans permutation), les éléments diagonaux constituent les points bien classés alors les éléments anti-diagonaux représentent les erreurs de l'algorithme. Malheureusement, il arrive que l'on ait une permutation des classes dûe à une initialisation différente ou la nature aléatoire de la méthode (Ex. : type Monté Carlo). La même

matrice que précédemment peut ainsi être réécrite :

	Π_1	Π_2	Π_3	Π_4	Total
C_1	93	1	4	3	101
C_2	5	75	4	23	107
C_3	2	32	1	86	121
C_4	0	1	141	5	147
Total	100	109	150	117	476

avec une permutation des classes (3) et (4). La vraisemblance obtenue pour les deux classifications est la même mais une interprétation automatique des résultats nécessite de détecter et de corriger les éventuelles permutations. L'information contenue dans la matrice de confusion peut être réduite sous la forme :

	Correct $= \sigma(C_i)$	Correct $\neq \sigma(C_i)$
C_i	α	β
$\bar{C}_i$	γ	δ

Pour une permutation σ donnée, on définit les variables :

L'exhaustivité (le rappel)	$R = \frac{\alpha}{\alpha+\gamma}$ si $\alpha+\gamma > 0$ sinon $R = 1$
La pertinence	$P = \frac{\alpha}{\alpha+\beta}$ si $\alpha+\beta > 0$ sinon $P = 1$
Le fallout	$f = \frac{\beta}{\beta+\delta}$ si $\beta+\delta > 0$ sinon $f = 1$
La précision	$Acc = \frac{\alpha+\delta}{n}$ où n désigne la taille de l'échantillon
L'erreur	$Err = \frac{\beta+\gamma}{n}$

Par la suite, on peut définir la mesure F telle que :

$$F(\sigma) = \frac{2R(\sigma)P(\sigma)}{R(\sigma) + P(\sigma)}$$

Pour la trouver la permutation optimale σ^*, il suffit alors de chercher celle qui maximise F. Ce critère est issu de l'informatique documentaire [Rij79].

Annexe B

Quelques fonctions annexes

La distribution Gamma

Paramètres	α et λ tels que $\alpha > 0$ et $\lambda > 0$
Densité de probabilité	$f(x;\alpha,\lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{\lambda x},$ pour $x > 0$
Moments	$E(X) = \frac{\alpha}{\lambda}, Var(X) = \frac{\alpha}{\lambda^2}$
Cas particuliers	$Exp(\lambda) = Gam(1,\lambda)$
	$\chi^2(\nu) = Gam(\frac{\nu}{2}, \frac{1}{2})$

La fonction Gamma

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx, \alpha > 0$$

Valeurs particulières :

$$\Gamma(1) = 1,$$
$$\Gamma(n+1) = n! \text{ pour n entier naturel,}$$
$$\Gamma(\frac{1}{2}) = \sqrt{(}\pi)$$

La fonction Digamma

$$\psi(\alpha) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$$

Bibliographie

[Aka74] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC–19 :716–723, 1974.

[ALB97] A. Amar, N.T. Labzour, and A. Bensaid. Semi-supervised hierarchical clustering algorithms. In *Sixth Scandinavian Conference on Artificial Intelligence*, pages 232–239, Helsinki, Finland, August 18-20 1997.

[Amb96] C. Ambroise. *Approche probabiliste en classification automatique et contraintes de voisinage*. PhD thesis, Université de Technologie de Compiègne, France, 1996.

[BB98] A. Bensaid and J. Bezdek. Semi-supervised point-prototype clustering. *Int'l Journal of Pattern Recognition and Artificial Intelligence*, 12(5), 1998.

[BCGW81] J. Bezdek, C. Coray, R. Gunderson, and J. Watson. Detection and characterization of cluster substructure. ii. fuzzy c-varieties and convex combinations thereof. *SIAM Journal of Applied Mathematics*, 40 :358–372, 1981.

[BD75] J. Bezdek and J. Dunn. Optimal fuzzy partitions : A heuristic for estimating the parameters in a mixture of normal distributions. *IEEE Transactions on Computers*, 24(8) :835–838, 1975.

[BD99] K. Bennett and A. Demiriz. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, 11 :368–374, 1999.

[Bel61] R. Bellman. *Adaptive Control Processes : A Guided Tour*. Princeton University Press, Princeton, New Jersey, U.S.A., 1961.

[Bez87] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, (second edition) edition, 1987.

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. CRC Press, 1984.

[BG98] C. Biernacki and G. Govaert. Choosing models in model-based clustering and discriminant analysis. Technical Report RR-3509, Inria, Institut National de Recherche en Informatique et en Automatique, 1998.

[BHBC96] A. Bensaid, L. Hall, J. Bezdek, and L. Clarke. Partially supervised clustering for image segmentation. *Pattern Recognition*, 29 :859–871, 1996.

[Bie97] C. Biernacki. *Choix de modèles en classification.* PhD thesis, Université de technologie de Compiègne, 1997.

[Bie99] C. Biernacki. XEMgaus : Software for model-based cluster and discriminant analysis, 1999. http ://citeseer.nj.nec.com/435204.html.

[BKKP99] J. Bezdek, J. Keller, R. Krishnapuram, and N. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.* Kluwer Academic Publishers, 1999.

[BM98a] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. University of California, Irvine, Dept. of Information and Computer Sciences, http ://www.ics.uci.edu/∼mlearn/MLRepository.html.

[BM98b] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. 1998.

[Bob88] L. Bobrowski. Feature selection based on some homogeneity coefficient. In *In Proceedings of Ninth International Conference on Pattern Recognition*, pages 544–546, 1988.

[Box79] G. Box. *Robustness in the strategy of scientific model building.* eds. R. L. Launer, and G. N. Wilkinson, New York : Academic Press, 1979.

[Boz90] H. Bozdogan. On the information-based measure of covariance complexity and its application to the evaluation of multivariate linear models. *Communications in Statistics : Theory and Methods (A)*, 19(1) :221–278, 1990.

[BR93] J. Banfield and A. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49 :803–821, 1993.

[Bru78] P. Brucker. On the complexity of clustering problems. *Optimierung und Operations Research, In R. Henn, B. Korte and W. Oletti (Eds). Lecture notes in Economics and Mathematical Systems*, 1978.

[Bur98] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167, 1998.

[CCD95] G. Celeux, D. Chauveau, and J. Diebolt. On stochastic versions of the EM algorithm. Technical Report RR-2514, Inria, Institut National de Recherche en Informatique et en Automatique, 1995. http ://citeseer.nj.nec.com/celeux95stochastic.html.

[CCFM99] G. Celeux, S. Chretien, F. Forbes, and A. Mkhadri. A component-wise EM algorithm for mixtures. Technical Report RR-3746, Inria, Institut National de Recherche en Informatique et en Automatique, 1999.

[CD85] G. Celeux and J. Diebolt. The SEM algorithm : a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational Statistics Quarterly*, 2(1) :73–92, 1985.

[CG92] G. Celeux and G. Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, 14 :315–332, 1992.

[CG95] G. Celeux and G. Govaert. Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5) :781–793, May 1995.

[CMU+98] R. Cole, J. Mariani, H. Uszkoreit, G. Varile, A. Zaenen, and A. Zampolli. *Survey of the State of the Art in Human Language Technology.* Cambridge University Press, 1998.

[CP95] J.P. Cocquerez and S. Philipp. *Analyse d'images : filtrage et segmentation.* Eds. Masson, 1995.

[CPL94] CPLEX Optimization, Inc. Using the CPLEX callable library, 1994. http ://www.ilog.com/products/cplex/.

[Dan98] M. Van Dang. *Classification de données spatiales : Modèles probabilistes et critères de partitionnement.* PhD thesis, Université Technologique de Compiègne, Décembre 1998.

[Dav91] R. Davé. Characterization and detection of noise in clustering. *Pattern Recognition*, 12(11) :657–664, 1991.

[Dav92] R. Davé. Generalized fuzzy c-shells clustering and detection of circular and elliptical boundaries. *Pattern Recognition*, 25 :713–721, 1992.

[DB92] R. Davé and K. Bhaswan. Adaptive fuzzy c-shells clustering and detection of ellipses. *IEEE Transactions on Neural Network*, 3(5) :643–662, 1992.

[DBE99] A. Demiriz, K. Bennett, and M. Embrechts. Semi-supervised clustering using genetic algorithms. *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 809–814, 1999.

[DH73] R. Duda and P. Hart. *Pattern Recognition and Scene Analysis.* John Wiley and Sons, 1973.

[DH83] D. Donoho and P. Huber. *The notion of breakdown point*, pages 57–184. Wadsworth, 1983. In a Festschrift for Erich L. Lehmann (P. J. Bickel, K. A. Doksum, and J. L. Hodges, Jr., eds.).

[DK97] R. Davé and R. Krishnapuram. Robust clustering methods : A unified view. *IEEE Transactions on Fuzzy Systems*, 5(2) :270–293, 1997.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society series B*, 39 :1–38, 1977.

[DM93] B. Dubuisson and M-H. Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognition*, 26 :155–165, 1993.

[Dui76] R. Duin. On the choice of smoothing parameters for parzen estimators of probability density functions. *IEEE Transactions on Computers*, 25(11) :1175–1179, 1976.

[Dun74] J. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3) :32–57, 1974.

[Fis36] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2) :179–188, 1936.

[FK99] H. Frigui and R. Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(5) :450–465, May 1999.

[FM00] G. Fung and O. Mangasarian. Data selection for support vector machine classifiers. In *Knowledge Discovery and Data Mining*, pages 64–70, 2000.

[FR98a] C. Fraley and A. Raftery. How many clusters ? which clustering method ? answers via model-based cluster analysis. Technical Report 329, Seattle : Department of Statistics, University of Washington, 1998.

[Fré98b] C. Frélicot. On unifying probabilistic/fuzzy and possibilistic rejection-based classifiers. *In : Amin A., Pudil P.(eds), Lecture Notes in Computer Science 1451 : Advances in Pattern Recognition. Springer-Verlag*, pages 736–745, 1998.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition, Second Edition.* Academic Press, Boston, MA, 1990.

[GG89] I. Geva and A Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 11(7), 1989.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, 1979.

[GK79] D. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. *In Proceedings of the IEEE Conference on Decision and Control*, pages 761–766, 1979.

[Gol89] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[GS90] V. Govindan and A. Shivaprasad. Character recognition – a survey. *Pattern Recognition*, 23(7) :671–683, 1990.

[Ha97] T-M. Ha. The optimum class-selective rejection rule. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19 :608–615, 1997.

[Hab98] M. Habib. Polycopié du cours d'algorithmique. DEA d'informatique, Université de Montpellier II, 1998.

[Har58] H. Hartley. Maximum likelihood estimation from incomplete data. *Biometrics*, 14 :174–194, 1958.

[HL01] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. *A paraître dans IEEE Transactions on Neural Networks*, 2001.

[Hol75] J. Holland. *Adaptation in natural artificial systems.* University of Michigan Press, Ann Arbor, 1975.

[Hsi98] P. Hsieh. *Classification of high dimensional data.* PhD thesis, School of electrical and computer engineering, Purdue University, West lafayette, Indiana 47907-1285, May 1998.

[Hub81] P. J. Huber. *Robust Statistics.* John Wiley, New York, 1981.

[HW77] P. Holland and R. Welsch. Robust regression using iteratively reweighted least squares. *Commun. Statis.-Theor. Meth., A6(9)*, pages 813–828, 1977.

[JBW81] R. Gunderson J. Bezdek, C. Coray and J. Watson. Detection and characterization of cluster substructure : I. linear structure : fuzzy c-lines. *SIAM Journal of Applied Mathemathics*, 40(2) :339–357, 1981.

[JDM00] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition : A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1) :4–38, 2000.

[Joa98] T. Joachims. Text categorization with support vector machines : learning with many relevant features. *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 1398 :137–142, 1998.

[JOEM00] F. Jouzel, C. Olivier, and A. El Matouat. Choix du nombre de composantes d'un mélange gaussien par critères d'information. *12 ème Congrès Francophone AFRIF-AFIA, RFIA 2000*, 1 :149–155, 2000.

[KJK96] R. Krishnapuram and J. J. Keller. The possibilistic c-means algorithm : Insights and recommendations. *IEEE Trans. Fuzzy Systems*, 4(3) :385–393, 1996.

[KK93] R. Krishnapuram and J.M Keller. A possibilistic approach to clustering. *IEEE Trans. on Fuzzy Systems*, 1(2) :98–110, 1993.

[KRT99] G. King, O. Rosen, and M. Tanner. Binomial-beta hierarchical models for ecological inference. *Sociological Methods and Research*, 28 :61–90, 1999.

[Kul59] S. Kullback. *Information Theory and Statistics.* Wiley, New York, 1959.

[LBB98] N.T. Labzour, A. Bensaid, and J Bezdek. Improved semi-supervised point-prototype clustering algorithms. *IEEE Int'l Conf. on Fuzzy Systems , IEEE World Congress on Computational Intelligence*, pages 1383–1387, May 1998.

[LFHM01] D.B. Goldgof L.M. Fletcher-Heath, L.O. Hall and F. Reed Murtagh. Automatic segmentation of non-enhancing brain tumors in magnetic resonance images. *Artificial Intelligence in Medicine*, 21(A) :43–63, 2001.

[Lon98] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8) :983–1001, 1998.

[LR87] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data.* John Wiley and Sons, 1987.

[LSH01] J. Larsen, A. Szymkowiak, and L. Hansen. Probabilistic hierarchical clustering with labeled and unlabeled data. *A paraître dans Int. Journal of Knowledge Based Intelligent Engineering Systems*, 2001.

[Mah36] P. Mahalanobis. On generalized distance in statistics. *Proceedings of the National Inst. Sci. (India)*, 12 :49–55, 1936.

[MK97] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* New York : Wiley., 1997.

[MP98] G. McLachlan and D. Peel. Robust cluster analysis via mixtures of multivariate t-distributions. *Lecture Notes in Computer Science*, 1451 :658–667, 1998.

[MP00] G. McLachlan and D. Peel. *Finite Mixture Models.* Eds Wiley, 2000. ISBN 0-471-00626-2.

[NJ95] T. Newman and A. Jain. A survey of automated visual inspection. *Computer Vision and Image Understanding*, 61 :231–262, 1995.

[OFG97] E. Osuna, R. Freund, and F. Girosi. Training support vector machines :an application to face detection, 1997.

[Par62] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33 :1065–1076, 1962.

[PD97] G. Phansalkar and R. Davé. On generating solid models of mechanical parts through fuzzy clustering. *Sixth IEEE International Conference on Fuzzy Systems (FUZZY-IEEE 97)*, I :225–230, 1997.

[Ped85] W. Pedrycz. Algorithms for fuzzy clustering with partial supervision. *Pattern Recognition Letters*, 3 :13–20, 1985.

[PM00] D. Peel and G. McLachlan. Robust mixture modelling using the t distribution. *Statistics and Computing*, 10(4) :339–348, Octobre 2000.

[Pos99] C. Posse. Hierarchical model-based clustering for large datasets. Technical Report 363, University of Washington, Novembre 1999.

[PV98] M. Pontil and A. Verri. Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6) :637–646, 1998.

[PW97] W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. *IEEE Transactions on Systems, Man, and Cybernetics, Part B : Cybernetics*, 27(5) :787–795, 1997.

[Qui93] J. Quinlan. *C 4.5 : Programs for machine learning*. Morgan Kaufmann, 1993.

[Rey83] W. Rey. *Introduction to robust and quasi-robust statistical methods*. Springer-Verlag, New York, 1983.

[Rij79] C. Van Rijsbergen. *Information Retrieval*. Butterworths, second edition, 1979.

[RL87] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection*. New York : John Wiley., 1987.

[Row98] S. Roweis. EM algorithms for PCA and SPCA. 10, 1998.

[Rus69] E. Ruspini. A new approach to clustering. *Information and Control*, 8 :338–353., 1969.

[Sam69] J. W. Sammon, Jr. A non-linear mapping for data structure analysis. *IEEE Transactions on Computers*, 18 :401–409, 1969.

[Sap90] G. Saporta. *Analyse des données et statistique*. Eds. Technip, 1990.

[Sar98] Warren S. Sarle. Prediction with missing inputs, 1998. http ://citeseer.nj.nec.com/sarle98prediction.html.

[Sch78] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6 :461–464, 1978.

[SJF01a] C. Saint-Jean and C. Frélicot. *An hybrid parametric model for semi-supervised robust clustering*. Conference on Recent Developments in Mixture Modelling (Mixtures 2001), 2001.

[SJF01b] C. Saint-Jean and C. Frélicot. *Une méthode paramétrique et robuste de classification semi-supervisée avec rejet*, pages 85–100. Dans les actes de la Conférence sur l'APprentissage (CAP' 2001), 2001.

[SJFV00a] C. Saint-Jean, C. Frélicot, and B. Vachon. *Clustering with EM : complex models vs. robust estimation*, pages 872–881. In proceedings of SPR 2000 : F. J. Ferri, J. M. Inesta, A. Amin, and P. Pudil (Eds.). Lectures Notes in Computer Science 1876, Springer-Verlag, 2000.

[SJFV00b] C. Saint-Jean, C. Frélicot, and B. Vachon. EM : Complexifier le modèle ou estimer de manière robuste. *12 ème Congrès Francophone AFRIF-AFIA, RFIA 2000*, 1 :139–148, 2000.

[SRI99] T. Steinherz, E. Rivlin, and N. Intrator. Off-line cursive script word recognition : A survey. *International Journal of Document Analysis and Recognition*, 2(2) :90–110, 1999.

[Ste00] M. Stephens. Dealing with label-switching in mixture models. *Journal of Royal Statistical Society series B*, 2000.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 14(3) :354–356, 1969.

[Sym81] M. Symons. Clustering criteria and multivariate normal mixtures. *Biometrics*, 37 :35–43, 1981.

[TJT96] O. Trier, A. Jain, and R. Taxt. Feature extraction methods for character recognition - a survey. *Pattern Recognition*, 29(4) :641–662, 1996.

[TK99] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press Inc. - ISBN 0-12-686140-4, 1999.

[Tru79] G. Trunk. A problem of dimensionality : A simple example. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1(3) :306–307, July 1979.

[UNGH99] N. Ueda, R. Nakano, Z. Ghahramani, and G. Hinton. SMEM algorithm for mixture models. *In Proceedings of Neural Information Processing Systems (NIPS)*, 1999.

[Vap82] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

[Vap95] V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, Heidelberg, DE, 1995.

[VK00] I. Shmulevich V. Katkovnik. Nonparametric density estimation with adaptive varying window size. *Conference on Image and Signal Processing for Remote Sensing VI, European Symposium on Remote Sensing*, 2000.

[War63] J. H. Ward, Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58 :236–244, 1963.

[WT90] G. Wei and M. Tanner. A monte carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85 :699–704, 1990.

[Zad65] L. Zadeh. Fuzzy sets. *Information and Control*, 8 :338–353, 1965.

[Zha95] Z. Zhang. Parameter estimation techniques : A tutorial with application to conic fitting. Technical Report RR-2676, Inria, Institut National de Recherche en Informatique et en Automatique, October 1995.

Résumé

L'apprentissage est une étape importante d'un processus de reconnaissance des formes pour la décision. On distingue généralement l'approche supervisée de l'approche non supervisée suivant que l'on dispose ou non d'une expertise des données. Dans ce travail, nous étudions le cas intermédiaire d'une classification semi-supervisée où l'on dispose d'un ensemble mixte de données numériques.
Certains éléments à traiter diffèrent du modèle a priori supposé des données et peuvent perturber le processus d'apprentissage. Les méthodes robustes de classification visent à limiter l'influence de ces données aberrantes soit en les modélisant explicitement, soit en utilisant des estimateurs robustes. La première partie de ce travail nous a permis d'étudier la notion de robustesse à travers divers algorithmes de classification. Un intérêt particulier est porté à l'utilisation des M-estimateurs de Huber dans le cadre de l'estimation par le principe du maximum de vraisemblance.
La seconde partie de cette étude est consacrée à l'état de l'art des principales méthodes de classification semi-supervisée. Nous montrons que celles-ci reposent sur la modification de la fonctionnelle réalisant la classification en introduisant un terme d'accord avec la mesure d'appartenance fixée par l'expert.
Sur la base de ces deux domaines, nous proposons un algorithme robuste de classification partiellement supervisée introduisant une option de rejet. Les classes sont modélisées par un mélange de deux composantes dont les paramètres sont estimées par un calcul itératif robuste. Le rejet est effectué par une fonction d'affectation produisant une classe additionnelle dédiée aux points aberrants. Les résultats obtenus sur divers jeux de données artificiels et réels nous ont permis de valider notre approche.

Mots-clés: Classification automatique, Robustesse, Semi-supervision, Algorithme EM, Rejet.

Abstract

Classifier design is a significant stage of a pattern recognition process. One generally distinguishes the supervised approach (classification) from the unsupervised one (clustering) according to whether expertise on data is available or not. In this work, we study the intermediate case of a semi-supervised clustering for mixed pool of numerical data. Whatever the approach, some elements, called outliers, differ from the a priori model for the data and can therefore disturb the clustering process. Robust clustering methods aim at limiting the influence of these outliers either by modelling them explicitly, or by using robust estimators.

In the first part of this work, we study the concept of robustness through various algorithms for clustering data. We focus on the use of so called M-estimators within the framework of estimation based on likelihood maximization. The second part of this study deals with a state of the art of semi-supervised clustering methods. We show that partial supervision is introduced by modifying the objective function with a term of agreement with respect to membership degrees or posterior probabilities fixed by the expert.

Finally, we propose a robust algorithm for clustering data in a partially supervised way. A reject option is introduced. Classes are modelled by a mixture of two components whose parameters are estimated through an iterative robust process. Rejection is achieved through assignment to an additional class dedicated to outliers. The proposed approach have been successfully applied on various artificial and real data sets.

Keywords: Clustering, Robustness, Semi-supervision, EM Algorithm, Reject option.