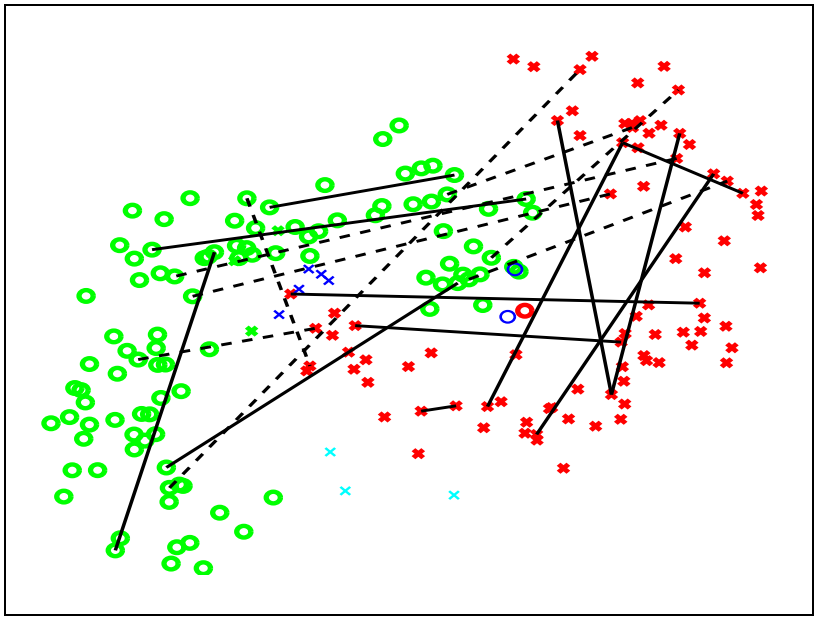
utc
Université de Technologie
Compiègne

par Violaine Antoine

Intégration de contraintes en classification automatique évidentielle

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC.



Soutenue le : 29 novembre 2011
Spécialité : Décision et Images

Intégration de contraintes en classification automatique évidentielle

Thèse soutenue le 29 novembre 2011 devant le jury composé de :

Mme.	Isabelle Bloch	Professeur, Télécom ParisTech, Paris	(Rapporteur)
Mr.	Olivier Colot	Professeur, Université Lille 1, Lille	(Rapporteur)
Mme.	Marie-Hélène Masson	Enseignant-Chercheur, UTC, Compiègne	(Directeur de thèse)
Mr.	Benjamin Quost	Enseignant-Chercheur, UTC, Compiègne	(Co-directeur de thèse)
Mr.	Thierry Denœux	Professeur, UTC, Compiègne	(Président du jury)
Mr.	David Mercier	Enseignant-Chercheur, Université d'Artois	(Examinateur)

Remerciements

Tout d'abord, j'exprime ma profonde reconnaissance à mes directeurs de thèse, Marie-Hélène Masson et Benjamin Quost, pour leurs conseils, leur patience et leurs encouragements. Ils m'ont tous deux laissé une grande autonomie de travail tout en restant disponibles à mes questions. J'ai réellement apprécié de travailler avec eux. Je remercie également les chercheurs qui m'ont guidée ponctuellement vers des pistes intéressantes de recherche, notamment Thierry Denœux pour ses précieux conseils dans le domaine de la théorie des fonctions de croyance et Pierre Villon pour son aide en optimisation.

Je souhaite remercier les membres de mon jury : Madame Isabelle Bloch et Monsieur Olivier Colot pour avoir accepté de rapporter sur ce mémoire, Monsieur David Mercier pour avoir bien voulu assister à ma présentation et Thierry Denœx pour m'avoir fait l'honneur de présider ce jury. Je leur sais gré de leurs remarques et critiques constructives.

Mes remerciements vont également à tous ceux qui ont contribué au bon déroulement de ma thèse durant ces trois années : merci au personnel administratif, notamment Céline Ledent, Sabine Vidal et Nathalie Alexandre pour leur aide logistique et leur patience, merci au personnel de la cellule informatique, tout spécialement Véronique Moisan et Gildas Bayard, pour la transmission de leurs connaissances, merci aux enseignants, Véronique Cherfaoui, Harry Claisse, Stéphane Crozat, Mohamed Bouali et Sinan Hatahet, qui m'ont épaulée dans l'apprentissage de leur profession. J'adresse aussi une pensée à Corinne Boscolo, avec qui j'ai beaucoup apprécié de travailler.

Je tiens par ailleurs à remercier tous mes amis : mes amis de Compiègne qui m'ont soutenue dans les moments difficiles, notamment Aurore, Laura et Krystyna, ainsi que mes amis d'enfance qui m'ont permis de temps à autre de me détendre et de me ressourcer. Enfin, j'adresse ma profonde gratitude à ma famille pour son soutien constant.

Résumé

La classification automatique est l'une des branches de l'analyse de données qui vise à regrouper des objets similaires en classes. Quand elle n'est pas connue directement, la mesure de similarité entre objets se fonde sur une description des objets par des attributs le plus souvent numériques. Par essence, la recherche d'une structure de classes est faite de manière non supervisée, c'est-à-dire qu'elle est guidée uniquement par les caractéristiques des objets ou leur mesure de dissimilarité. Dans certaines applications cependant, une connaissance humaine supplémentaire sur les objets ou sur les classes en présence est disponible. Dans ce contexte, nous proposons de combiner deux concepts existants en classification automatique.

Le premier concept, la classification sous contraintes, consiste à introduire une connaissance a priori sous forme de contraintes sur la partition recherchée. Il existe différentes formes de contraintes à différents niveaux du modèle. Au niveau des objets par exemple, une contrainte "Must-Link" spécifie que deux objets doivent être dans la même classe et une contrainte "Cannot-Link" indique que deux objets ne doivent pas être dans la même classe. L'ajout de contraintes permet une amélioration sensible des résultats de classification.

Le second concept correspond à l'utilisation des fonctions de croyance — et notamment l'utilisation de la notion de partition crédale — en classification non supervisée. La notion de partition crédale généralise les notions de partitions dures et floues et permet en particulier de gérer les points aberrants, c'est-à-dire ceux qui n'appartiennent à aucune classe.

Nous introduisons dans cette thèse deux nouveaux algorithmes de classification sous contraintes en utilisant le cadre théorique des fonctions de croyance. L'un est dédié aux données de type individus-variables, l'autre aux données de dissimilarités. Leurs performances sont évaluées sur différents jeux de données synthétiques et réels.

Mots-clés :
classification automatique, classification sous contraintes, théorie des fonctions de croyances, partition crédale, contraintes Must-link et Cannot-link.

Abstract

Cluster analysis is one branch of data analysis that aims at grouping similar objects into cluster. When a dissimilarity measure between objects is not known a priori, it is based on the description of the objects which is usually composed of numerical attributes. In essence, the research of clusters structure is unsupervised. Indeed, it is only guided by the characteristics of the objects or by their dissimilarity measures. However, for some applications or in particular domains, it exists some extra knowledge on the objects or on the classes. In this framework, we propose to combine two concepts of clustering.

The first one, called constrained clustering, aims at introducing background knowledge in the form of constraints in order to guide the algorithm towards a desired solution. It exists different type of constraints, at different level. In the instance level for example, a Must-Link constraint specifies that two objects should be in the same class and a Cannot-Link constraint indicates that two objects are not in the same class. Adding constraints enable to perceptibly improve the accuracy of the classification.

The second concept corresponds to the use of belief functions — and particularly the use of some notion : the credal partition — in clustering. The notion of credal partition enhance the notions of hard and fuzzy partitions and enables to represents a wide range of situations concerning the class membership of an objet. For example, it handles particularly well ouliers.

We introduce in this thesis two new constrained clustering algorithms that use the framework of belief functions. The first one is dedicated to feature vector data and the second one to relational data. The results are evaluated on synthetic and real databases.

KEYWORDS :
clustering, constrained clustering, belief functions theory, credal partition, Must-Link and Cannot-Link constraints.

Publications

Revue Internationale

1. **V. Antoine**, B. Quost, M.-H. Masson, T. Denœux, *CECM : Constrained Evidential C-Means algorithm*, Computational Statistics and Data Analysis, 2011. A paraître.

Conférences Internationales

1. **V. Antoine**, B. Quost, M.-H. Masson, T. Denœux, *CECM : Adding pairwise constraints to evidential clustering*, Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Barcelona, Espagne, 2010.

2. **V. Antoine**, B. Quost, M.-H. Masson, T. Denœux, *Evidential clustering with constraints and adaptive metric*, Workshop on the theory of belief functions, Brest, France, 2010.

3. **V. Antoine**, B. Quost, M.-H. Masson, T. Denœux, *CEVCLUS : Constrained evidential clustering of proximity data*, Proceedings of the European Conference on Fuzzy Logic and Technology, Aix-Les-Bains, France, 2011.

Conférence Nationale

1. **V. Antoine**, B. Quost, M.-H. Masson, T. Denœux, *Algorithme évidentiel des c-moyennes avec contraintes*. Rencontres Francophones sur la Logique Floue et ses Applications (LFA), Annecy, France, 2009.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Introduction

Problématique

L'évolution rapide des moyens informatiques a contribué à faire exploser la quantité d'informations disponibles dans le monde. Autrefois réduites à un stockage sur papier, ces données, issues par exemple d'Internet, de la téléphonie ou de capteurs, sont maintenant placées dans des entrepôts numériques de plus en plus performants. Un individu ne pouvant traiter seul une telle quantité d'informations rapidement, il est de nos jours devenu nécessaire de les analyser automatiquement. L'analyse des données inclut en particulier les méthodes d'organisation automatique qui permettent à un individu d'avoir une vision globale d'un ensemble d'informations, dans une perspective d'aide à la décision. Par exemple dans le domaine de l'imagerie médicale, la segmentation de zones homogènes permet de mettre en évidence des régions pathologiques. De même, dans le domaine de la bioinformatique, la proposition de regroupement de gènes ayant les mêmes expressions peut permettre à un expert de mettre en lumière des interactions entre gènes. Dans le domaine de la mercatique, il est intéressant de découper le marché en sous-ensembles dont les individus réagissent de manière similaire. Cela permet par exemple de mieux cibler la clientèle en lui proposant des promotions adéquates.

Ces méthodes d'organisation automatique, ou plus communément nommées méthodes de classification automatique, permettent de constituer des groupes homogènes d'objets distincts les uns des autres. Dans le cas le plus simple, le partitionnement des objets est dur, c'est-à-dire que chaque objet est affecté à une et une seule classe. La recherche d'une description la plus riche possible de l'appartenances aux classes amène à utiliser des modèles mathématiques permettant de représenter divers concepts tels que l'incertitude et l'imprécision. Ainsi, la classification floue construit une partition basée sur les probabilités, ce qui permet de modéliser l'incertitude planant sur l'appartenance d'un objet à une classe. La classification évidentielle, introduite récemment, se fonde sur la théorie des fonctions de croyance [54]. De tels objets mathématiques, qui englobent les probabilités comme cas particulier, permettent d'exprimer tout type de connaissance, de l'ignorance totale à la certitude complète quant à l'appartenance d'un objet à une classe. De cette manière, un expert peut prendre lui même une décision par rapport aux objets pour lesquels l'algorithme de classification évidentiel émet des doutes.

Sans autre connaissance a priori que les données issues d'entrepôts numériques, la classification automatique peut se révéler une tâche difficile. En effet, plusieurs questions apparaissent dès lors qu'un utilisateur veut organiser ses données : comment définir la notion d'homogénéité entre objets ? Comment définir les différentes

classes ? Comment juger une classification par rapport à une autre ? Un faible apport de connaissances supplémentaires pourrait permettre de répondre à la plupart de ces questions. Il existe souvent des connaissances a priori liées au domaine de l'expert. Par exemple en imagerie médicale, les médecins peuvent déjà connaître la forme globale de la zone pathologique qu'ils recherchent. En bioinformatique, une vaste base de données de gènes apparentés et déjà connus est disponible. Ainsi, à l'aide de ces connaissances, les experts peuvent guider les algorithmes de classification vers la structure sous-jacente aux données qui leur semble la plus appropriée. La simplicité d'obtention de connaissances supplémentaires et l'amélioration des résultats de classification ont révélé l'intérêt d'intégrer une telle supervision faible à un algorithme de classification automatique. Les connaissances a priori, ajoutées sous forme de contraintes, donnent naissance à de nouveaux algorithmes nommés algorithmes de classification sous contraintes [4]. Il existe de nombreuses façons d'exprimer ces contraintes, et ce tant au niveau des classes qu'au niveau des objets. Deux types de contraintes sont fréquemment utilisés au niveau des objets [64] : la première contrainte, nommée Must-Link, indique que deux objets doivent être dans la même classe. La seconde, nommée Cannot-Link, spécifie que deux objets ne sont pas dans la même classe.

L'approche que nous proposons consiste à coupler la notion de classification évidentielle avec la notion de classification automatique sous contraintes. Par cette stratégie novatrice, nous cherchons à améliorer les résultats de classification. Deux algorithmes de classification évidentielle sont utilisés : le premier, nommé ECM [44], est une variante de l'algorithme des centres mobiles se fondant sur les fonctions de croyance. Le second, nommé EVCLUS [21], est un algorithme traitant en entrée de données de type relationnel, c'est-à-dire de données sous forme de dissimilarité entre objets. Ces deux méthodes sont modifiées afin de prendre en compte des contraintes de type Must-Link et Cannot-Link.

Organisation du mémoire

Les deux premiers chapitres de ce mémoire constituent un état de l'art des notions de classification sous contraintes et évidentielle. Le premier chapitre est consacré à la classification automatique sous contraintes. Il est tout d'abord dédié aux différentes approches de la classification automatique, puis aux différents types de contraintes qu'il est possible de construire à partir de connaissances a priori. Les différentes manières d'intégrer ces contraintes dans un algorithme de classification automatique sont ensuite développées. Le second chapitre fait un état de l'art de la classification automatique dans le cadre des fonctions de croyance. Le cadre théorique des fonctions de croyance est dans un premier temps présenté. Les méthodes de classification évidentielle ECM et EVCLUS sont ensuite décrites.

Les contributions de cette thèse sont présentées dans les chapitres 3 et 4. Le chapitre 3 présente un algorithme sous contraintes dérivé de ECM alors que le chapitre 4 décrit un nouvel algorithme sous contraintes construit à partir de EVCLUS. Dans chacun de ces chapitres, la manière dont sont intégrées les contraintes est tout

d'abord exposée. Différents résultats obtenus sur des jeux de données de la littérature sont ensuite présentés. Ils permettent d'analyser et d'interpréter de manière approfondie le comportement des deux nouveaux algorithmes.

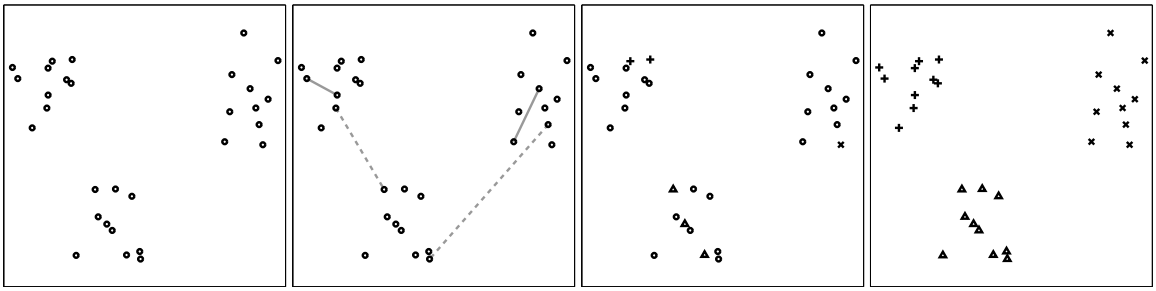
Classification automatique sous contraintes

La classification automatique, ou clustering en anglais, est une méthode d'analyse des données qui cherche à constituer des groupes d'objets tels que les objets soient les plus similaires au sein d'un groupe et que les groupes soient les plus dissemblables entre eux. Les objets, appelées aussi individus ou points, peuvent être décrits par des caractéristiques ou par une matrice de dissimilarités entre individus. Il existe deux grandes familles de méthodes en classification automatique : la première, qui correspond aux méthodes de partitionnement, définit l'appartenance de chaque objet aux classes au moyen d'une partition. La seconde famille comprend les méthodes hiérarchiques qui génèrent des suites de partitions emboîtées.

La classification automatique cherche à découvrir la structure intrinsèque des données à partir des caractéristiques des objets ou des dissimilarités entre individus, sans autre connaissance a priori. Or une faible supervision peut guider l'algorithme vers une solution désirée ou corriger les imperfections d'une partition. Prenons par exemple une base de données composée d'images correspondant à des portraits : la classification peut être effectuée selon les émotions issues du visage (tristesse, joie, étonnement...) ou selon l'identité des personnes. Ainsi, des informations supplémentaires sont nécessaires pour mener l'algorithme vers la partition désirée. L'ajout de connaissances a priori peut alors être vue comme une contrainte sur la structure recherchée : on parle de classification sous contraintes. Il existe plusieurs types de contraintes dans la littérature. Les plus connues sont les contraintes Must-Link et Cannot-Link qui définissent si deux objets sont ou pas dans la même classe.

Le terme de classification semi-supervisée est parfois employé dans la littérature pour faire référence à la classification sous contraintes. Il englobe cependant d'autres notions comme celle de la discrimination semi-supervisée, qui consiste à apprendre une règle de classement à partir de données étiquetées (c'est-à-dire d'individus dont les classes sont déjà connues) et non étiquetées. Afin d'éviter toute ambiguïté, nous parlerons uniquement de classification sous contraintes.

Ces différentes notions sont illustrées par la figure 1.1. Nous nous intéressons dans ce mémoire uniquement aux algorithmes de classification automatique et sous contraintes.



(a) (b) (c) (d)

Figure 1.1 – La classification automatique classique (a) utilise uniquement les caractéristiques des données. La classification sous contraintes (b) utilise un petit nombre de connaissances sur les objets à classer (comme par exemple des contraintes par paires) ; la discrimination semi-supervisé (c) considère que seul un certain nombre d'étiquettes sont connues et l'apprentissage supervisé (d) nécessite un ensemble d'apprentissage.

Ce chapitre débute par une présentation de la classification non supervisée. Plus particulièrement, les algorithmes de classification automatique sur lesquels se basent les méthodes de classification évidentielle du chapitre 2 sont détaillés. Dans une seconde partie, les méthodes de classification automatique sous contraintes et leurs concepts sous-jacents sont analysés.

1.1 La classification automatique

1.1.1 Les différentes approches

Les méthodes de classification automatique peuvent être organisées selon les types de données disponibles en entrée, les différentes techniques de regroupement et la structure de classification recherchée en sortie.

Données en entrée

Les données en entrée (ou objets notées $O = \{o_1, \ldots, o_n\}$) peuvent avoir de nombreuses formes, les plus classiques étant la forme vectorielle et la forme relationnelle.

La forme vectorielle Ce type de données, utilisé dans la plupart des algorithmes de classification automatique, se présente sous la forme d'une matrice multivariée $\mathbf{X}$ de taille $(n \times p)$ caractérisant les valeurs des attributs des n objets à classifier :

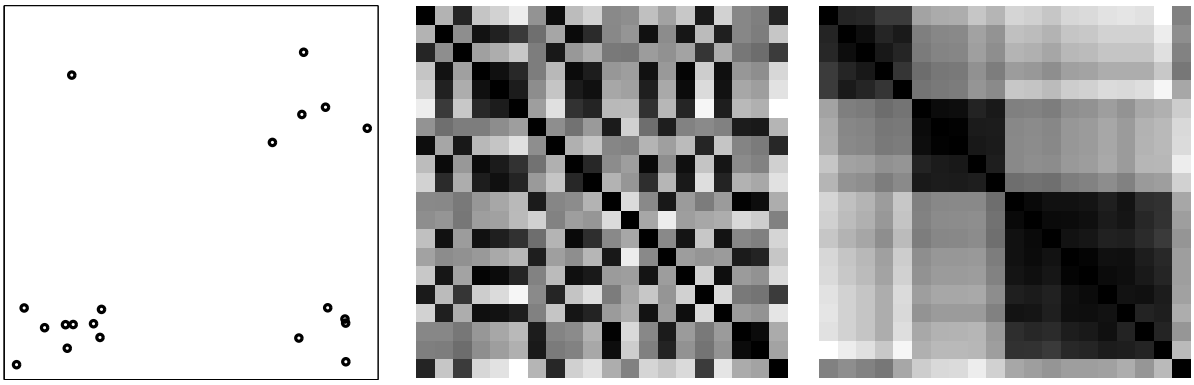
$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{n1} & \ldots & \ldots & x_{np} \end{pmatrix}.$$

Le vecteur $\mathbf{x}_i$ inclu dans l'espace $\mathcal{X} = \mathbb{R}^p$ décrit les caractéristiques de l'objet i et x_{ij} représente le j^{eme} attribut de l'objet i. Par abus de langage, on pourra utiliser la notation $\mathbf{x}_i$ pour désigner l'objet i. La matrice $\mathbf{X}$ peut être convertie en une matrice de dissimilarités par différentes procédures telles que le calcul de la distance Euclidienne pour chaque paire de points.

La visualisation de l'ensemble d'une base de données est importante car elle permet de mieux comprendre la structure des données. Dans le cas de données vectorielles, il est possible d'utiliser l'analyse en composante principale (ACP) qui consiste à transformer les attributs possiblement liés entre eux afin de les rendre indépendants les uns des autres. Les deux nouveaux attributs les plus informatifs sont alors utilisés pour visualiser les données en deux dimensions. Un pourcentage d'information (ou pourcentage d'inertie) apporté par cette représentation des données peut être calculé et permet de connaître le degré d'exactitude de la représentation par rapport à la réalité.

La forme relationnelle Ce type de données, moins informatif que le précédent, est fréquent dans le domaine des sciences empiriques telles que la biochimie, la psychologie ou encore la linguistique. Il consiste en une matrice $\mathbf{D}$ de taille $(n \times n)$ représentant les dissimilarités inter-objets. Ces dissimilarités correspondent très souvent à une mesure de distance.

Le moyen le plus simple de visualiser des données relationnelles est de représenter les dissimilarités sous la forme d'une image en niveau de gris (cf. figures 1.2(a) et 1.2(b)). Chaque pixel situé aux coordonnées (i, j) représente la dissimilarité d_{ij} entre les objets i et j. Ce pixel est noir si $d_{ij} = 0$ et devient de plus en plus clair si la dissimilarité augmente. Afin d'améliorer la visualisation des dissimilarités, il est ensuite possible d'utiliser une méthode de réorganisation des objets nommée VAT [5] qui ordonne les objets selon une comparaison de leur dissimilarité (cf. figure 1.2(c)).



(a) (b) (c)

Figure 1.2 – Exemples de représentation de données relationnelles. Des données bidimensionnelles sont générées automatiquement et montrent clairement trois classes et un point isolé (a). En calculant les distances Euclidienne entre chaque points, il est possible de créer des données relationnelles et de les présenter sous forme d'une image (b). Afin de mieux cerner la structure des données, l'algorithme VAT est appliqué (c). Il est maintenant aisé de constater qu'il existe trois groupes et un objet isolé.

Avant d'utiliser un algorithme de classification non-supervisée, il est possible de procéder à un prétraitement des données. Ce dernier dépend du type de données d'entrée et de l'algorithme de classification sélectionné. Il peut correspondre à une modification de l'espace de représentation (cf. les méthodes à noyau partie 1.1.3), à

un débruitage, une normalisation ou une extraction de caractéristiques pertinentes des données, etc.

Stratégies de regroupement automatique

Les algorithmes de classification automatique peuvent être divisés en deux familles, qui elles-mêmes peuvent être séparées en sous-ensembles (cf. figure 1.3). La première famille correspond aux méthodes paramétriques, qui imposent aux objets de former une structure basée sur des distributions de probabilité. Ces méthodes de classification, nommées modèles de mélange, utilisent en amont des méthodes d'estimation des paramètres. L'algorithme d'Espérance-Maximisation (plus communément appelé EM) [18] est l'une des méthodes d'estimation la plus connue. La seconde stratégie rassemble les méthodes non-paramétriques. Parmi celles-ci se trouvent principalement les méthodes basées sur des prototypes et les méthodes de classification hiérarchique.

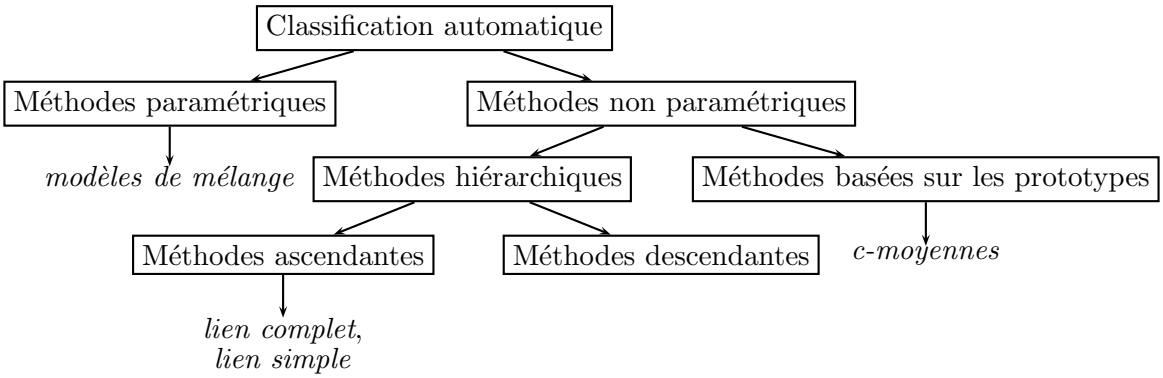


Figure 1.3 – Techniques de classification automatique.

Les méthodes basées sur des prototypes sont des méthodes pour lesquelles chaque classe est représentée par un prototype. Ainsi, dans le cas de l'algorithme des centres mobiles (plus communément appelé c-moyennes ou k-means en anglais) [43], un prototype correspond au centre de gravité des objets appartenant à une classe.

Les méthodes de classification hiérarchique [24] sont des méthodes de graphes qui s'appuient sur la hiérarchisation de suites de classes emboîtées. Le principe, dans le cas de la classification hiérarchique descendante, est de partir d'une classe composée de l'ensemble des données pour la diviser successivement en sous-groupes jusqu'à obtenir n classes contenant chacune un unique objet. À l'inverse, la classification hiérarchique ascendante procède par fusions successives des groupes déjà existants. Il existe de nombreux critères de fusion. Le lien simple par exemple définit la distance entre deux groupes comme la plus courte des distances entre un objet d'un groupe et un objet de l'autre groupe. Une fusion des deux classes pour lesquelles cette distance est minimale est alors possible. Cela permet d'obtenir une partition avec des classes très éloignées les unes des autres. Le lien complet sélectionne les paires de points entre deux classes les plus éloignées, calcule les distances sous-jacentes puis fusionne les deux classes ayant une distance minimale. Les classes obtenues sont plus équilibrées et plus compactes (bien que plus proche les unes des autres) que les classes obtenue avec la technique du lien simple.

Structure de classification

Les méthodes de classification automatique fournissent en sortie soit un partitionnement des objets, soit un ensemble de partitions emboîtées (c'est le cas par exemple de la classification hiérarchique). Nous nous intéressons ici plus particulièrement aux méthodes qui génèrent un partitionnement. Une partition peut être dure, c'est-à-dire que chaque objet est affecté à une unique classe. Cependant ce choix de partition est peu réaliste car il ne peut pas détecter les objets ayant des caractéristiques communes avec des objets de différentes classes. Ainsi une frontière floue semble plus adaptée pour représenter l'ambiguïté quant à l'appartenance de certains objets à plusieurs classes. La partition floue, notée $\mathbf{U} = (u_{ik})$, est une matrice pour laquelle chaque objet i appartient à chaque classe k avec un degré d'appartenance u_{ik}. Cette matrice, de taille $(n \times c)$ telle que c correspond au nombre de classes, vérifie les propriétés suivantes :

$$u_{ik} \geq 0 \quad \forall i, k, \tag{1.1}$$

$$\sum_{k=1}^{c} u_{ik} = 1. \tag{1.2}$$

D'autres types de partitionnement comme la partition possibiliste ou la partition crédale existent, et permettent une analyse particulière des résultats (la partition crédale sera présentée dans la partie 2.2.1).

1.1.2 L'algorithme des centres mobiles et ses dérivés

Nous présentons maintenant l'algorithme des c-moyennes et ses principales variantes. Ces algorithmes sont en effet à la base de nombreuses versions décrites dans la seconde partie de ce chapitre et sont à l'origine des contributions présentées dans le chapitre 2. Ils ont la particularité d'être rapides et simples à implémenter. Ils ont donc été appliqués avec succès dans de nombreux domaines tels que la vision par ordinateur, la géostatique, l'astronomie, etc.

Algorithme des c-moyennes

L'algorithme des c-moyennes (CM, ou k-means en anglais) cherche à minimiser l'inertie intra-classe, c'est-à-dire la distance entre les objets d'une classe et leur prototype associé [43]. Pour cela, Lloyd propose une heuristique (cf. algorithme 1.1) qui optimise alternativement la partition dure des données $P = (P_1, \ldots, P_c)$ composée de c classes et les prototypes définis par les centres de gravité $V = (\mathbf{v}_1, \ldots, \mathbf{v}_c)$ [42]. Si chaque objet i est représenté par un vecteur de caractéristiques $\mathbf{x}_i$, alors $d(\mathbf{x}_i, \mathbf{v}_k)$ (noté plus simplement d_{ik}) définit la distance entre un objet i et le centre de la classe k. L'algorithme minimise donc la fonction objectif suivante :

$$J_{CM}(P, \mathbf{V}) = \sum_{i=1}^{n} \sum_{\mathbf{x}_i \in P_k} d^2(\mathbf{x}_i, \mathbf{v}_k). \tag{1.3}$$

Il faut noter que la distance d_{ik} est dans la plupart des applications la distance Euclidienne.

Algorithme 1.1 : Pseudo-code de l'algorithme des c-moyennes "générique"

Entrées : Données $\mathbf{X}$ sous forme vectorielle
Sorties : Partition dure $P = (P_1, \ldots, P_c)$
début
  Initialisation aléatoire de c prototypes $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_c)$
  tant que *non convergence* **faire**
    Affecter chaque objet au prototype le plus proche :

$$P_k = \{x_i : k = \arg\min_{k'} d(\mathbf{x}_i, \mathbf{v}_{k'})\}$$

    Calculer les nouveaux prototypes V^t tels que :

$$\mathbf{v}_k = \frac{1}{|P_k|} \sum_{\mathbf{x}_i \in P_k} \mathbf{x}_i \quad \text{avec } |P_k| \text{ le nombre d'objets dans la classe } k$$

fin

Plusieurs conditions d'arrêt peuvent être utilisées. Il est ainsi possible de mesurer la stabilité de la partition entre les itérations t et $t-1$ afin d'arrêter l'algorithme lorsque la partition n'évolue plus. De manière équivalente, il est possible de se baser sur le critère (1.3) ou sur l'évolution des centres. Le choix de l'approche n'a pas d'importance car ces dernières sont dépendantes l'une de l'autre. En effet si la variation de la partition diminue, alors les variations des prototypes et de la fonction objectif diminue, et vice versa.

Algorithme des c-moyennes floues

L'algorithme des c-moyennes floues, plus communément nommé fuzzy c-means (FCM) en anglais, a été développé par Dunn [23] puis amélioré par Bezdek [6]. Basé sur l'algorithme des c-moyennes, il s'applique à trouver une partition floue $\mathbf{U}$ qui minimise la fonction objectif suivante :

$$J_{FCM}(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{n} \sum_{k=1}^{c} u_{ik}^{\beta} d_{ik}^2, \tag{1.4}$$

sous les contraintes (1.1) et (1.2). Le coefficient $\beta > 1$ est un paramètre fixe réglant la dureté de la partition : plus β a une valeur élevée et plus la partition $\mathbf{U}$ s'approchera d'une partition dure. Le critère J_{FCM} est minimisé de la même manière que pour l'algorithme des c-moyennes, c'est-à-dire en alternant l'optimisation de la partition floue et des centres de gravité. Les équations de mise à jour des degrés d'appartenance et des prototypes trouvés à l'aide des conditions de Kuhn et Tucker sont les suivantes :

$$u_{ik} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d_{ik}}{d_{il}} \right)^{\frac{2}{\beta-1}}} \quad \forall i \in \{1, \ldots, n\}, k \in \{1, \ldots, c\}, \tag{1.5}$$

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n} u_{ij}^{\beta} \mathbf{x}_i}{\sum_{i=1}^{n} u_{ij}^{\beta}} \quad \forall k \in \{1, \ldots, c\}. \tag{1.6}$$

L'utilisation de la distance Euclidienne $d_{ik} = (\mathbf{x}_i - \mathbf{v}_k)^\top (\mathbf{x}_i - \mathbf{v}_k)$ rend l'algorithme adapté à la reconnaissance de classes sphériques de même volume. D'autres distances peuvent être utilisées, comme par exemple la distance de Mahalanobis :

$$d_{ik}^2 = (\mathbf{x}_i - \mathbf{v}_k)^\top \mathbf{S} (\mathbf{x}_i - \mathbf{v}_k), \tag{1.7}$$

où $\mathbf{S}$ est une matrice semi-définie positive qui correspond à l'inverse de la matrice de variance-covariance des données. Cette distance permet la recherche de classes de formes ellipsoïdales de même orientation et même taille.

Algorithme des c-moyennes floues et métrique adaptative

Gustafson et Kessel proposent une variante de FCM qui permet d'adapter automatiquement à chaque classe une mesure de distance [33]. La métrique utilisée dans l'algorithme est définie, pour tout $i \in \{1, \ldots n\}$ et $k \in \{1, \ldots c\}$ par :

$$d_{ik} = (\mathbf{x}_i - \mathbf{v}_k)^\top \mathbf{S}_k (\mathbf{x}_i - \mathbf{v}_k), \tag{1.8}$$

où $\mathbf{S}_k$ représente la matrice de distance associée à la classe k. La minimisation du nouveau critère J_{GK} (cf. équation (1.10)) est donc maintenant effectuée en alternant l'optimisation des degrés d'appartenance, des centres de gravité et des matrices $\mathbf{S}_k$. Notons que les formules de mise à jour des degrés d'appartenance et des prototypes sont identiques dans J_{FCM} et J_{GK}, car elles ne dépendent pas de la métrique.

Une solution triviale à ce problème d'optimisation consiste à sélectionner les matrices $\mathbf{S}_k$ nulles. De cette manière, les distances d_{ik} sont nulles et par conséquent le critère J_{GK} est lui aussi égal à 0. Pour éviter cette solution aberrante, Gustafson et Kessel proposent d'ajouter une contrainte de volume à chaque matrice :

$$det(\mathbf{S}_k) = \varrho_k. \tag{1.9}$$

Le paramètre ϱ_k correspond à une constante dont le choix dépend uniquement de la connaissance a priori du problème. Il est donc généralement fixé à 1 lorsqu'aucune information n'est disponible. Le problème d'optimisation est donc formulé de la manière suivante :

$$J_{GK}(\mathbf{U}, \mathbf{V}, \mathbf{S}) = \sum_{i=1}^{n} \sum_{k=1}^{c} u_{ik}^{\beta} d_{ik}^2, \tag{1.10}$$

sous les contraintes (1.1), (1.2) et (1.9).

Algorithme des c-moyennes floues et gestion de données bruitées

Un inconvénient majeur de l'algorithme des c-moyennes floues est qu'il aboutit parfois à des résultats contre-intuitifs en présence de données bruitées ou aberrantes. C'est pourquoi Davé propose une variante de la méthode en introduisant une classe dédiée à ces objets atypiques [14]. La méthode, appelée "noise-clustering" (NC), consiste à minimiser la fonction objectif suivante :

$$J_{NC}(\mathbf{U},\mathbf{V}) = \sum_{i=1}^{n}\sum_{k=1}^{c} u_{ik}^{\beta} d_{ik}^{2} + \sum_{i=1}^{n} \rho^{2} u_{i*}^{\beta}, \tag{1.11}$$

où u_{i*} représente le degré d'appartenance de l'objet i à la classe consacrée aux objets bruités et ρ correspond à une distance fixe entre les données et l'isobarycentre de cette classe particulière. Les paramètres du modèle sont calculés en minimisant (1.11) sous les contraintes (1.1) et (1.2).

Algorithme des c-moyennes floues et choix du nombre de classes

En classification automatique, le choix du nombre de classes est un problème délicat. L'algorithme des c-moyennes, dont la fonction objectif dépend de ce paramètre, ne fait pas exception à cette difficulté. Le nombre de classes doit être fixé a priori : si il est correct, il permet généralement d'obtenir une partition pertinente. En revanche, si ce nombre de classes est sous ou sur-estimé, le résultat final sera peu représentatif de la structure sous-jacente des données. Différentes analyses en prétraitement, telles que des tests statistiques ou la "méthode du coude", permettent d'estimer la valeur de ce paramètre. Une autre stratégie consiste à modifier le critère à optimiser, afin de le rendre indépendant du choix du nombre de classes. Dans ce cadre, Frigui et Krishnapuram considère le nombre de classes c comme un paramètre à optimiser [26]. Le minimum global de J_{FCM} est alors atteint quand c est égal au nombre d'individus n. Les auteurs proposent alors d'ajouter un second terme pour lequel le minimum global est atteint quand tous les points sont dans une classe unique (et donc toutes les autres classes sont vides). Combiner les deux termes à l'aide d'un hyper-paramètres ζ permet de trouver une partition floue qui minimise la distance intra-classe et le nombre de classes. l'algorithme, nommée Competitive-Agglomeration (CA), minimise la fonction objectif suivante :

$$J_{CA}(\mathbf{U},\mathbf{V}) = \sum_{i=1}^{n}\sum_{k=1}^{c} u_{ik}^{\beta} d_{ik}^{2} + \zeta \sum_{k=1}^{c} \left(\sum_{i=1}^{n} u_{ik} \right)^{2}, \tag{1.12}$$

sous contraintes (1.1) et (1.2).

L'algorithme CA sur-estime à l'initialisation le nombre de classes et réduit ensuite cette valeur en ne faisant survivre que les classes ayant un nombre important d'objets. Ainsi, après chaque mise à jour de la partition floue, la méthode calcule le nombre estimé d'objets de chaque classe k sous la forme $N_k = \sum_{i=1}^{n} u_{ik}$ et compare cette valeur à un seuil ε. Si $N_k < \varepsilon$, alors la classe k est supprimée.

L'algorithme des c-moyennes et ses dérivées décrits ci-dessus utilisent comme données d'entrée des données vectorielles. La variante suivante permet d'adapter FCM afin traiter une matrice de dissimilarités D en entrée.

Algorithme des c-moyennes floues relationnel

Pour pouvoir utiliser des données relationnelles, Hathaway & al proposent de modifier la fonction objectif de FCM (1.4) de sorte qu'elle n'implique plus le calcul d'une dissimilarité entre les objets et les centres des classes [36] mais s'appuie seulement sur les dissimilarités entre les paires d'objets. La nouvelle dissimilarité considérée correspond alors à la dissimilarité entre les paires d'objets et la fonctionnelle minimisée s'écrit :

$$J_{RFCM}(\mathbf{U}) = \frac{1}{2}\sum_{k=1}^{c}\left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}u_{ik}^{\beta}u_{jk}^{\beta}d_{ij}^{2}}{\sum_{i=1}^{n}u_{ik}^{\beta}}\right), \tag{1.13}$$

sous contraintes (1.1) et (1.2), où d_{ij} désigne la dissimilarité entre l'objet i et l'objet j.

Les auteurs montrent que si la matrice de dissimilarités correspond à une distance Euclidienne alors la minimisation du critère équivaut à l'optimisation de J_{FCM}. Dans le cas où D se rapporte à une autre distance, le résultat de la classification peut être aberrant. Pour pallier ce problème, Hathaway et Bezdek proposent une variante nommée NERCFM [35]. Dans cette dernière, une étape est ajoutée à l'algorithme de base : à chaque itération, la matrice de dissimilarités D est modifiée de façon à être Euclidienne.

1.1.3 Les méthodes à noyaux

Les algorithmes de classification automatique comme ceux qui viennent d'être présentés ne permettent pas d'obtenir des classes non linéairement séparables. Afin de rendre possible la détection de classes de formes variées, les méthodes à noyaux plongent les données initiales $\mathbf{X} \subseteq \mathcal{X}$ dans un espace $\mathcal{H}$ de plus grande dimension dans lequel le problème devient linéaire.

Un certain nombre d'algorithmes de classification automatique peuvent se reformuler uniquement à l'aide de produits scalaires. Utiliser une méthode à noyaux consiste alors à transformer les données initiales à l'aide d'une fonction Φ puis à calculer les produits scalaires entre les objets pris deux à deux. Cette démarche peut s'avérer fastidieuse, voire impossible dans le cas où $\mathcal{H}$ correspond à espace de dimension infinie. Une astuce (nommée "kernel trick" en anglais) consiste à trouver une fonction κ (cf. définition (1.1)) qui calcule les produits scalaires dans l'espace $\mathcal{H}$ de manière implicite (c'est-à-dire à partir de l'espace $\mathcal{X}$). L'astuce du noyau est illustrée par l'exemple 1.1.

Définition 1.1. *(Fonction Noyau) Une fonction noyau est une application* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *telle que :*

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) =< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >, \quad \forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}, \tag{1.14}$$

avec $\Phi : \mathcal{X} \to \mathcal{H}$*, où* $\mathcal{H}$ *est un espace muni d'un produit scalaire* $< \cdot, \cdot >$*.*

Exemple 1.1. *(Noyau polynomial de degré 2) Soient* $\mathbf{x}_i = (x_{i1}, x_{i2})$ *et* Φ *une fonction telle que* $\Phi : \mathbf{x}_i \to \Phi(\mathbf{x}_i) = (x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2})$*. Le produit scalaire* $< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$ *peut donc être calculé :*

$$\begin{aligned} < \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) > &= < (x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2}), (x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2}) > \\ &= x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} \\ &= (x_{i1}x_{j1} + x_{i2}x_{j2})^2 \\ &= < \mathbf{x}_i, \mathbf{x}_j >^2 \\ &= \kappa(\mathbf{x}_i, \mathbf{x}_j). \end{aligned} \tag{1.15}$$

La figure 1.4 montre l'impact de la transformation par Φ *sur un jeu de données composé de deux classes. La frontière de décision, initialement non linéaire (figure 1.4(a)), devient linéaire après augmentation de la dimension de l'espace (figure 1.4(b)). La fonction noyau* $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ *permet de calculer l'ensemble des produits scalaires entre les images des objets dans l'espace* $\mathcal{H}$ *(figure 1.4(c)) directement à partir de leur représentation dans l'espace initial* $\mathcal{X}$*. La transformation* Φ *n'a donc pas besoin d'être explicite.*

La distance inter-objet d'un jeu de données constitue généralement le principal moyen de grouper ces objets selon leur similarité et dissimilarité. Cette distance, qui est généralement la distance Euclidienne, doit pouvoir être calculée à partir de l'espace $\mathcal{X}$ tout en faisant référence au nouvel espace $\mathcal{H}$.

Définition 1.2. *(Distance Euclidienne dans l'espace* $\mathcal{H}$*) Cette distance peut être calculée à partir de la fonction noyau :*

$$\begin{aligned} \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 &= < \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) > -2 < \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) > + < \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) > \\ &= \kappa(\mathbf{x}_i, \mathbf{x}_i) - 2\kappa(\mathbf{x}_i, \mathbf{x}_j) + \kappa(\mathbf{x}_j, \mathbf{x}_j). \end{aligned} \tag{1.16}$$

Si le noyau correspond à une fonction qui considère uniquement une distance, alors les algorithmes de classification automatique dont les données en entrée sont relationnelles peuvent utiliser eux aussi l'astuce du noyau.

Le choix d'un noyau peut s'avérer très difficile. Chacun a en effet ses particularités et peut détecter des formes de classe particulières. Les noyaux les plus fréquemment utilisés sont les noyaux polynomial, gaussien et sigmoïde.

Exemple 1.2. *(Noyau gaussien) Le noyau gaussien est défini par la fonction suivante :*

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{d_{ij}^2}{2\sigma^2}\right), \tag{1.17}$$

où d_{ij} *représente la distance Euclidienne entre* $\mathbf{x}_i$ *et* $\mathbf{x}_j$ *et* σ *la largeur de bande.*

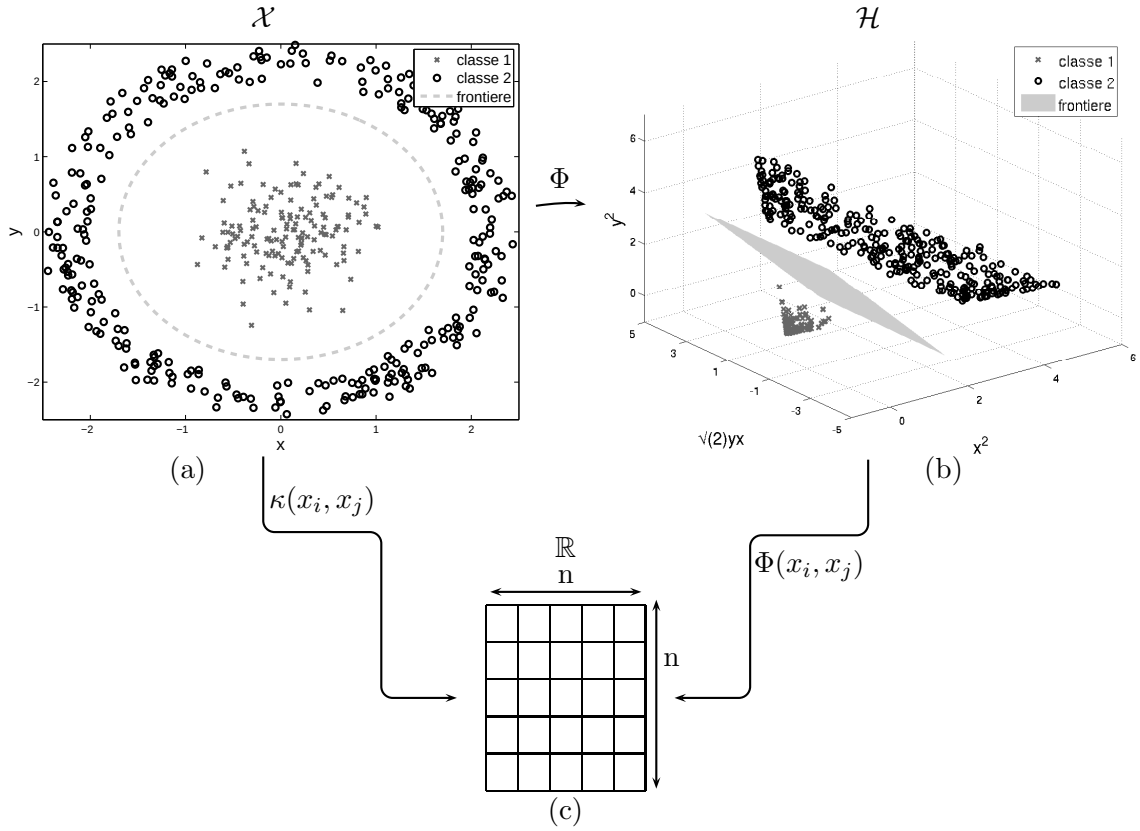


Figure 1.4 – Exemple de modification de la représentation des données avec un noyau polynômial de degré 2. Les classes dans l'espace d'origine (a) sont séparées par une frontière de décision ellipsoïdale. La transformation de cet espace (b) permet de définir une frontière linéaire. L'ensemble des produits scalaires résultant de l'espace transformé (c) peut être directement calculé à partir de l'espace initial.

1.2 La classification automatique sous contraintes

Nous nous intéressons ici à la classification automatique sous contraintes [4]. Dans ce cadre, la connaissance a priori disponible peut être exploitée à différents niveaux et peut être exprimée par différents types de contraintes. La connaissance a priori peut ainsi porter sur le modèle : Zhong et Ghosh proposent un algorithme qui équilibre le nombre d'objets par classe [74] et Gondek et Hofmann décrivent une méthode qui spécifie les solutions non désirées [30]. Il est également possible d'introduire des contraintes au niveau des classes, par exemple en spécifiant une distance maximale intra-classe et une distance minimale inter-classe [15]. Enfin au niveau des objets, des prédicats (tels que "$\mathbf{x}_i$ est plus proche de $\mathbf{x}_j$ que de $\mathbf{x}_l$") [53] ou des contraintes entre des paires objets [64] peuvent être formulées et intégrées dans le processus de détermination des classes.

Les contraintes entre des paires d'objets constituent un type de connaissance a priori très répandu dans le domaine de la classification automatique semi-supervisée [70, 73]. Elles sont divisées en deux types : une contrainte dite Must-Link spécifie que deux objets doivent être dans la même classe et une contrainte dite Cannot-Link indique que deux objets doivent être dans une classe différente [1]. De telles contrain-

tes ont été introduites de différentes manières dans des algorithmes de classification automatique, généralement en ajoutant un terme de pénalité à la fonction objectif ou en modifiant les distances entre les objets. Dans la suite, nous noterons $\mathcal{M}$ et $\mathcal{C}$ les ensembles de contraintes Must-Link et Cannot-Link, respectivement.

De nombreuses applications utilisant ce type de contraintes ont récemment vu le jour. Ainsi, en biomédecine, les spécialistes cherchent à classer des gènes à partir de leurs expressions. Ils ont à leur disposition une vaste base de données comprenant des gènes apparentés déjà découverts. La classification automatique sous contraintes permet donc d'exploiter cette connaissance a priori sous forme de contraintes Must-Link / Cannot-Link et améliore les résultats finaux [73]. D'autres applications ont été proposées, telles que l'identification de personnes sur une vidéo de surveillance [2], l'annotation d'images [50], la détection de routes par GPS [64], l'étude du langage naturel [11], etc.

Les algorithmes présentés ci-dessous utilisent des contraintes Must-Link et Cannot-Link. Un tableau comparatif est disponible à l'annexe A. Dans un premier temps, nous présentons les méthodes employées en prétraitement d'algorithmes classiques de classification automatique. Ces méthodes modifient la métrique de manière à prendre en compte les contraintes dans la classification. Dans une seconde partie, un ensemble de méthodes dérivées de l'algorithme des centres mobiles est étudié. Ces méthodes ajoutent un terme de pénalité à la fonction objectif afin de permettre la prise en compte des contraintes. Dans certains cas, elles peuvent aussi adapter automatiquement leur métrique en fonction des contraintes et de la structure intrasèque des données.

1.2.1 Les méthodes de modification de la métrique en prétraitement

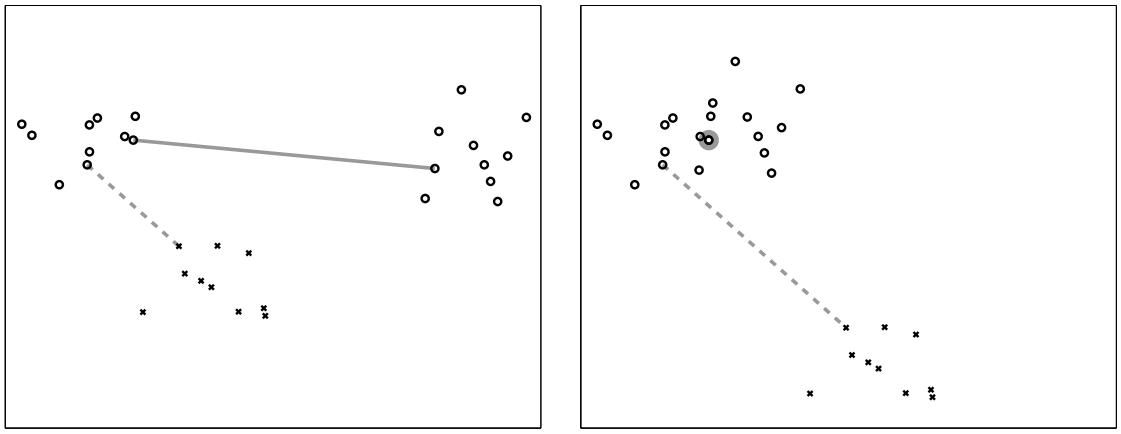
Les méthodes transformant la métrique initiale d'un jeu de données en fonction de contraintes Must-Link / Cannot-Link sont nombreuses. Parmi celles-ci, différentes approches se distinguent.

Approche relationnelle simple

La première approche, dite relationnelle car elle utilise uniquement une matrice de dissimilarités, consiste à déformer cette matrice de manière à respecter les contraintes. Ces dernières, considérées comme des informations spatiales, sont caractérisées par les deux propositions suivantes [38] :

- Si deux objets i et j sont liés par une contrainte Must-Link alors leur dissimilarité d_{ij} doit être faible. Par conséquent, tous les points proches de l'objet i doivent être proches de l'objet j et vice versa (cf. figure 1.5).
- Inversement, si deux objets i et j sont liés par une contrainte Cannot-Link alors leur dissimilarité d_{ij} doit être élevée. Ainsi, tous les points proches de l'objet i doivent être éloignés de l'objet j et vice versa (cf. figure 1.5).

Afin de respecter au mieux les inégalités triangulaires, Klein propose un algorithme en deux temps nommé Constrained Complete-Link (CCL)[38]. Tout d'abord, un graphe valué est construit à partir de la matrice des distances : chaque sommet



(a) Espace initial (b) Espace transformé

Figure 1.5 – Les données, présentées dans l'espace initial (a), doivent être séparées en deux classes. Les symboles (croix et cercles) indiquent la classe réelle des objets et le trait continu (respectivement discontinu) entre deux objets correspond à une contrainte Must-Link (respectivement Cannot-Link). La figure (a) montre clairement trois groupes. La contrainte Must-Link (respectivement Cannot-Link) entre deux objets de deux groupes différents permet de transformer l'espace (figure (b)) afin de rapprocher (respectivement d'éloigner) ces groupes.

représente alors un objet et chaque arc entre deux sommets la distance entre deux objets. Afin de respecter les contraintes Must-Link, la valeur des arcs entre les paires d'objets concernés par ce type de contrainte est fixée à 0. L'algorithme de Floyd-Warshall [12], qui permet de déterminer tous les plus courts chemins d'un graphe, est ensuite exécuté pour restaurer les inégalités triangulaires. Il contribue à la propagations des contraintes Must-Link aux points non contraints. Dans un deuxième temps, la distance entre les paires d'objets contraints par un Cannot-Link est fixée à une grande valeur (par exemple à $\max_{i,j} d_{ij} + 1$). Le rétablissement de l'inégalité triangulaire pour cette nouvelle matrice étant un problème difficile, les auteurs proposent de propager les contraintes et de classifier les données simultanément. Pour cela, ils utilisent la méthode de classification hiérarchique ascendante avec le critère d'aggrégation du lien complet.

Apprentissage d'une distance de Mahalanobis

L'algorithme Distance Metric Learning (DML) propose de chercher une métrique $\mathbf{S}$ de telle sorte que la distance entre deux objets soit minimale s'ils sont contraints par un Must-Link et maximale s'ils sont contraints par un Cannot-Link [67]. La fonction objectif suivante doit alors être minimisée :

$$\begin{aligned} J_{DML}(\mathbf{S}) = & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{S}(\mathbf{x}_i - \mathbf{x}_j), \\ \text{s.c.} & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{S}(\mathbf{x}_i - \mathbf{x}_j) \geq 1. \end{aligned} \tag{1.18}$$

Notons que la matrice $\mathbf{S}$, qui caractérise la métrique, peut être vue comme une matrice inverse de variance-covariance. Xing propose deux modifications de la fonction objectif selon la forme de cette matrice. En effet, si $\mathbf{S}$ est une matrice diagonale,

l'optimisation est rapide mais la relation des attributs entre eux n'est pas considérée : les éléments non diagonaux fixés à 0, les attributs sont supposés indépendants les uns des autres. Inversement, une matrice pleine implique une optimisation coûteuse en temps, mais exploite toutes les caractéristiques d'une matrice de variance-covariance.

Approche par noyau

Cette approche consiste à calculer une fonction noyau qui satisfasse les contraintes par paires. La fonction noyau doit trouver un nouvel espace qui minimise (respectivement maximise) la distance entre les objets contraints par un Must-Link (respectivement Cannot-Link). Cependant la prise en compte des contraintes seules dans le calcul d'une fonction noyau peut entraîner un bouleversement irrémédiable et non désiré de la structure initiale des données, notamment dans le cas d'un nombre peu important de contraintes. Par conséquent la plupart des algorithmes considèrent aussi la préservation de la structure des données dans leur fonction objectif [37, 41].

Il existe une grande variété de fonctions noyau permettant de prendre en compte des contraintes par paires. Dans le cadre d'une approche relationnelle, Yan et Domeniconi proposent de trouver la largeur de bande σ d'un noyau gaussien par la minimisation de la fonction objectif suivante [69] :

$$J_{KG}(\sigma) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_\sigma^2 - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_\sigma^2, \tag{1.19}$$

avec

$$\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_\sigma^2 = 2 - 2\exp\left(\frac{-d_{ij}^2}{2\sigma^2}\right). \tag{1.20}$$

Afin d'obtenir une solution plus souple, il est possible de considérer une fonction noyau composée d'une combinaison linéaire de noyaux gaussiens. Les paramètres à optimiser sont alors les largeurs de bande de chaque noyau gaussien. Notons de plus que comme cet algorithme ne prend en compte que les contraintes Must-Link et Cannot-Link pour trouver un noyau. Il améliore donc les résultats d'une classification uniquement si le nombre de contraintes est suffisamment important.

1.2.2 Les méthodes dérivées des centres mobiles

La classification automatique sous contraintes basée sur l'algorithme des centres mobiles a fait l'objet de nombreuses études. Les méthodes les plus connues et les plus proches des algorithmes que nous avons développés (voir chapitres 3 et 4) sont présentées ci-dessous.

COP-kmeans

L'algorithme COP-kmeans [64], qui correspond à une variante des c-moyennes, est l'une des premières méthodes développées pour la classification par contraintes. À ce titre, il sert souvent de référence dans la littérature. La différence entre cet algorithme et la méthode de base réside dans la manière d'affecter les objets aux classes. En effet dans COP-kmeans, chaque objet $\mathbf{x}_i$ est affecté à la classe dont le

centre de gravité est le plus proche de l'objet, à la condition que cette affectation ne viole pas de contrainte (cf. algorithme 1.2).

Algorithme 1.2 : Pseudo-code de l'algorithme des COP-kmeans

Entrées : Données $\mathbf{X}$ sous forme vectorielle, ensembles $\mathcal{M}$ des contraintes Must-Link et $\mathcal{C}$ des contraintes Cannot-Link.
Sorties : Partition dure $P = (P_1, \ldots, P_c)$
début
  Initialisation aléatoire de c prototypes $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_c)$
  tant que *non convergence* **faire**
    Affecter chaque objet $\mathbf{x}_i$ au prototype $\mathbf{v}_k$ le plus proche telle que RESPECT_CONTRAINTES($\mathbf{x}_i$,P_k,$\mathcal{M}$,$\mathcal{C}$)=VRAI
    si *Il existe un objet qui ne peut pas être affecté* **alors**
      échec de l'algorithme : retourner P=()
    Calculer les nouveaux prototypes $\mathbf{V}$
fin
fonction RESPECT_CONTRAINTES(o_i,P_k,$\mathcal{M}$,$\mathcal{C}$)
début
  pour *chaque* $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$ **faire**
    si $\mathbf{x}_j \notin P_k$ **alors** retourner faux
  pour *chaque* $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$ **faire**
    si $\mathbf{x}_j \in P_k$ **alors** retourner faux
  sinon retourner vrai
fin

L'algorithme COP-kmeans cherche une partition respectant toutes les contraintes sans retour possible vers une partition précédente. Or cette manière de procéder entraîne des problèmes de faisabilité : si il existe du bruit dans les contraintes, alors certaines d'entre elles peuvent être contradictoires, ce qui rend impossible le respect total des contraintes. Dans le cas où il n'existe pas de bruit, une solution n'est pas toujours évidente à obtenir, notamment pour un nombre important de contraintes. En effet le bon respect des contraintes dépend de l'initialisation des prototypes et de l'ordre dans lesquelles les objets sont affectés à une classe. Pour pallier ce problème, de nouvelles modifications de COP-kmeans ont été proposées. L'une d'entre elles consiste à assouplir la condition d'affectation de COP-kmeans afin de respecter le maximum de contraintes sans pour autant imposer le respect de toutes les contraintes [15]. Une autre étend cette idée pour rendre la méthode robuste aux contraintes bruitées [48].

PC-kmeans

L'algorithme PC-kmeans [3] ajoute deux termes à la fonction objectif des c-moyennes (voir équation (1.3)) pour pénaliser les solutions qui ne respectent pas les contraintes Must-Link et Cannot-Link :

$$J_{PCKM}(P, \mathbf{V}) = J_{CM}(P, \mathbf{V}) + \sum_{\substack{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}, \\ \mathbf{x}_i \in P_k, \mathbf{x}_j \in P_l, l \neq k}} \omega_{ij} + \sum_{\substack{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}, \\ \mathbf{x}_i, \mathbf{x}_j \in P_k}} \overline{\omega}_{ij}, \tag{1.21}$$

où ω_{ij} (respectivement $\overline{\omega}_{ij}$) représente le coût de violer une contrainte Must-Link (respectivement Cannot-Link). Les poids ω_{ij} et $\overline{\omega}_{ij}$, fixés a priori, spécifient un compromis entre le respect des contraintes et le respect de la structure des données. Si ces poids sont égaux à 0, alors PC-kmeans est équivalent à l'algorithme des c-moyennes. Si ces poids sont élevés, alors les contraintes doivent absolument être respectées et à l'instar de COP-kmeans, il n'existe plus de garantie d'obtenir une solution au problème.

MPC-kmeans [7]

L'algorithme PC-kmeans repose sur la distance Euclidienne. Bilenko et Basu proposent, de la même manière que Gustafson et Kessel [33], une variante de la méthode nommée MPC-kmeans, qui permet d'adapter la distance aux données et aux contraintes. À chaque classe k est attribuée une matrice $\mathbf{S}_k$ permettant de calculer la distance de Mahalanobis $d_{ik}^2 = (\mathbf{x}_i - \mathbf{v}_k)^\top \mathbf{S}_k (\mathbf{x}_i - \mathbf{v}_k)$ d'un point $\mathbf{x}_i$ au centre $\mathbf{v}_k$. L'ensemble des matrices $\mathbf{S} = \{\mathbf{S}_1, \ldots, \mathbf{S}_c\}$ doivent alors être déterminés conjointement à la partition et aux centres en optimisant le critère suivant :

$$\begin{aligned} J_{MPCKM}(P, \mathbf{V}, \mathbf{S}) \;=\; & \sum_{i=1}^{n} \sum_{\mathbf{x}_i \in P_k} \left(d_{ik}^2 - \log(\det(\mathbf{S}_k)) \right) \\ & + \sum_{\substack{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}, \\ \mathbf{x}_i \in P_k, \mathbf{x}_j \in P_l, l \neq k}} \omega_{ij} f_M(\mathbf{x}_i, \mathbf{x}_j) + \sum_{\substack{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}, \\ \mathbf{x}_i, \mathbf{x}_j \in P_k}} \overline{\omega}_{ij} f_C(\mathbf{x}_i, \mathbf{x}_j). \end{aligned} \tag{1.22}$$

Le terme $\log(\det(\mathbf{S}_k))$ est une contrainte de volume permettant d'éviter des solutions aberrantes telles que l'inverse d'une matrice de variance-covariance nulle. Deux objets $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$ distants l'un de l'autre et dont la contrainte est violée doivent générer une plus forte pénalité que deux objets dans la même situation mais proches l'un de l'autre. Autrement dit, une forte distance sur une contrainte Must-Link non respectée signifie que la métrique n'est pas adaptée et qu'elle doit être modifiée. Bilenko et Basu définissent donc la fonction f_M de la manière suivante :

$$\begin{aligned} & f_M(\mathbf{x}_i, \mathbf{x}_j) = \tfrac{1}{2} \left((\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{S}_k (\mathbf{x}_i - \mathbf{x}_j) + (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{S}_l (\mathbf{x}_i - \mathbf{x}_j) \right), \\ & \forall i, j \in \{1, \ldots n\}, \quad k, l \in \{1, \ldots c\}, \quad \mathbf{x}_i \in P_k, \quad \mathbf{x}_j \in P_l, \end{aligned} \tag{1.23}$$

avec $P = (P_1, \ldots, P_c)$ la partition dure.

De même, la pénalité pour deux points proches de violer une contrainte Cannot-Link doit être plus forte que pour deux points éloignés. Pour cela, la fonction f_C est définie par :

$$\begin{aligned} & f_C(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_{i'} - \mathbf{x}_{j'})^\top \mathbf{S}_k (\mathbf{x}_{i'} - \mathbf{x}_{j'}) - (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{S}_k (\mathbf{x}_i - \mathbf{x}_j), \\ & \forall i, j \in \{1, \ldots n\}, \quad k \in \{1, \ldots c\}, k \neq l, \quad \mathbf{x}_i, \mathbf{x}_j \in P_k, \end{aligned} \tag{1.24}$$

avec $(\mathbf{x}_{i'}, \mathbf{x}_{j'})$ la paire de points les plus éloignés au sens de la métrique $\mathbf{S}_k$.

PCCA

L'algorithme PCCA [31] est un dérivé de l'algorithme des c-moyennes floues. Il ajoute un terme de pénalité à la fonction objectif (1.12) de l'algorithme CA (cf. paragraphe 1.1.2) :

$$J_{PPCA}(\mathbf{U},\mathbf{V}) = J_{CA}(\mathbf{U},\mathbf{V}) + \xi \left(\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} \sum_{k=1}^{c} \sum_{l=1,l\neq k}^{c} u_{ik}u_{jl} + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} \sum_{k=1}^{c} u_{ik}u_{jk} \right), \tag{1.25}$$

sous contrainte (1.2). Le second terme de la fonction objectif (1.25) spécifie un coût pour chaque contrainte non respectée. Ainsi, la présence de deux objets contraints par un Must-Link dans des classes différentes est pénalisée par la somme des produits des degrés d'appartenance à des classes différentes. À l'inverse, la violation d'un Cannot-Link par une paire d'objets entraîne un coût égal à la somme des produits des degrés d'appartenance aux mêmes classes. Le paramètre ξ est une constante qui règle le compromis entre le premier terme de la fonction objectif et le terme de pénalité.

Il faut noter que la contrainte (1.1) n'est pas prise en compte dans l'optimisation de la fonction objectif. Les auteurs choisissent de vérifier son respect après minimisation du critère :

- si $u_{ij} < 0$ alors $u_{ij} \leftarrow 0$,
- si $u_{ij} > 1$ alors $u_{ij} \leftarrow 1$.

Cette correction fonctionne dans la mesure où pour une valeur de ξ correctement choisie, les degrés d'appartenance u_{ij} qui ne respectent pas la contrainte (1.1) sont tout de même proches de l'intervalle $[0;1]$.

SSCARD [25]

Afin de classifier des données dont seules les dissimilarités inter-objets et des contraintes Must-Link / Cannot-Link sont connues, Frigui et Krishnapuram proposent de modifier l'algorithme RFCM (cf. équation (1.13)). De la même manière que l'algorithme PCCA, ils ajoutent un terme à J_{RFCM} afin de pénaliser les contraintes non respectées. La fonction objectif est la suivante :

$$J_{CRFCM}(\mathbf{U},\mathbf{W}) = J_{RFCM}(\mathbf{U},\mathbf{W}) + \xi \left(\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} \sum_{k=1}^{c} \sum_{l=1,l\neq k}^{c} u_{ik}u_{jl} + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} \sum_{k=1}^{c} u_{ik}u_{jk} \right). \tag{1.26}$$

La méthode décrite dans [25] est un peu plus développée que celle présentée ci-dessus. En effet, les auteurs modifient aussi le terme J_{RFCM} afin de pouvoir prendre en compte plusieurs matrices de dissimilarités.

1.2.3 Apprentissage actif

Contre-performances en classification automatique sous contraintes

Sous l'hypothèse que la connaissance a priori injectée est pertinente, il est naturel de penser qu'un algorithme de classification sous contraintes donne de meilleurs résultats que le même algorithme sans cette connaissance. Cette intuition s'avère en réalité fausse : il existe des cas où injecter de la connaissance a priori fait chuter les performances de classification. Ainsi, en classification automatique semi-supervisée, Wagstaff teste différents ensembles de contraintes Must-Link et Cannot-Link sur les algorithmes COP-kmeans, PC-kmeans et MPC-kmeans et observe des contre-performances pour certains jeux de contraintes [63].

La possibilité d'une dégradation des performances amène à étudier le phénomène pour en comprendre la cause. Une analyse des contraintes par paires a été conduite sur ce sujet et deux propriétés importantes ont permis de définir deux mesures d'intérêt d'un ensemble de contraintes[16]. La première caractéristique d'une contrainte correspond à son utilité, c'est-à-dire à la quantité d'information que la contrainte apporte par rapport à ce qu'est capable de trouver l'algorithme de classification par lui-même. Il existe des contraintes plus ou moins informatives, comme le montre la figure 1.6. Une seconde caractéristique est sa cohérence par rapport aux autres contraintes étant donné une métrique. Ces mesures, bien qu'utiles dans la plupart des cas, ne peuvent cependant pas expliquer tous les comportements de classification, et certaines contraintes utiles et cohérentes peuvent tout de même induire une baisse des performances.

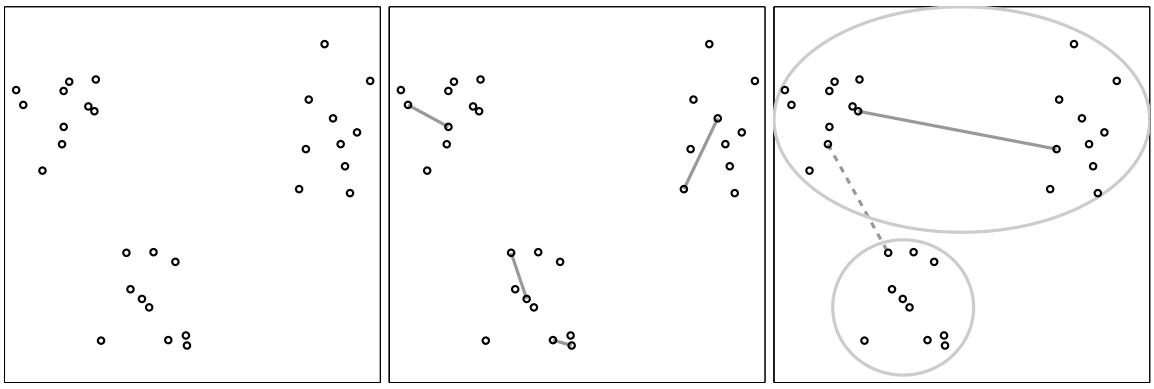


Figure 1.6 – Contraintes Must-Link et Cannot-Link pour un jeu de données (a) : certaines contraintes sont inutiles (b) alors que d'autres sont informatives (c) et mènent l'algorithme vers une solution désirée.

Dans le cas d'applications réelles, des experts ont souvent la possibilité de déterminer rapidement la contrainte existant entre deux objets. Cependant l'obtention aléatoire d'un nombre élevé de contraintes peut devenir aussi coûteux qu'inutile si ces contraintes s'avèrent être redondantes ou non-informatives. Ainsi, il est capital de choisir un ensemble réduit de contraintes qui aura le meilleur impact bénéfique sur la partition finale. Le schéma global, nommé apprentissage actif, consiste à demander à

un utilisateur quels types de contraintes relient différentes paires d'objets sélectionnées automatiquement. Les réponses sont collectées par l'algorithme et permettent à ce dernier de trouver une partition finale ou d'interroger de nouveau l'expert sur d'autres contraintes (cf figure 1.7).

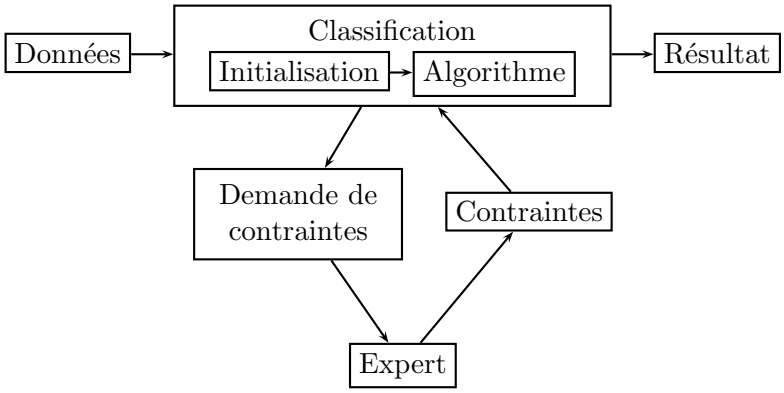


Figure 1.7 – Schéma de l'apprentissage actif.

La stratégie de sélection de contraintes diffère selon l'algorithme de classification automatique sous contraintes. En effet, une méthode dérivée des c-moyennes ne choisira pas de la même manière des paires d'objets qu'une méthode basée sur une classification hiérarchique.

Méthodes d'apprentissage actif en classification automatique sous contraintes

Plusieurs techniques d'apprentissage actif ont été proposées pour les méthodes dérivées des c-moyennes. Ces techniques sont présentes soit en prétraitement pour l'initialisation de l'algorithme, soit lors de l'exécution de celui-ci. Pour chaque méthode présentée ci-dessous, un nombre maximal Q de questions à poser à un expert est fixé.

La méthode Exploration/Consolidation : La fonction objectif de l'algorithme des c-moyennes ne permet pas une optimisation globale du critère. Les performances de l'algorithme dépendent donc fortement de l'initialisation des centres de gravité choisie. Afin d'éviter les minima locaux, une méthode consiste à réaliser plusieurs essais avec différentes initialisations et sauvegarder la partition associée à la plus faible valeur de J_{CM}. Cette méthode nécessite un nombre d'essais important, et peut donc s'avérer coûteuse en temps.

L'apprentissage actif permet de considérer une autre stratégie. Cette dernière, dite méthode d'Exploration/Consolidation, permet de choisir une initialisation unique et informative [3]. La phase d'exploration, comme son nom l'indique, explore l'ensemble des données dans le but de trouver au moins un objet par classe. Cette stratégie revient en fait à détecter des contraintes Cannot-Link. Soit $N = \{N_1, \ldots, N_c\}$ un ensemble de voisinages disjoints tel que N_k représente l'ensemble des points affectés de manière certaine à la classe k. À l'initialisation, un premier objet x_1 choisi aléatoirement est placé dans l'ensemble N_1. L'ensemble N est alors composé uniquement

de N_1. Le second objet x_2 sélectionné correspond à celui qui a une distance maximale avec le premier objet. Un expert est ensuite interrogé sur le lien existant entre ces deux objets. Si ceux-ci sont contraints par un Cannot-Link, alors x_2 est placé dans un nouvel ensemble N_2. Dans le cas contraire, x_2 est inclu dans N_1. Le troisième objet x_3 choisi est le point le plus éloigné de tous les objets déjà sélectionnés. Si pour un objet de chaque ensemble de N il existe avec x_3 une contrainte Cannot-Link, alors x_3 est un représentant d'une nouvelle classe et doit être placé dans un nouvel ensemble. Plusieurs objets sont testés de cette manière jusqu'à atteindre le quota de questions ou le nombre de classes c attendu (cf. algorithme 1.3).

Algorithme 1.3 : Pseudo-code de l'algorithme d'exploration des données

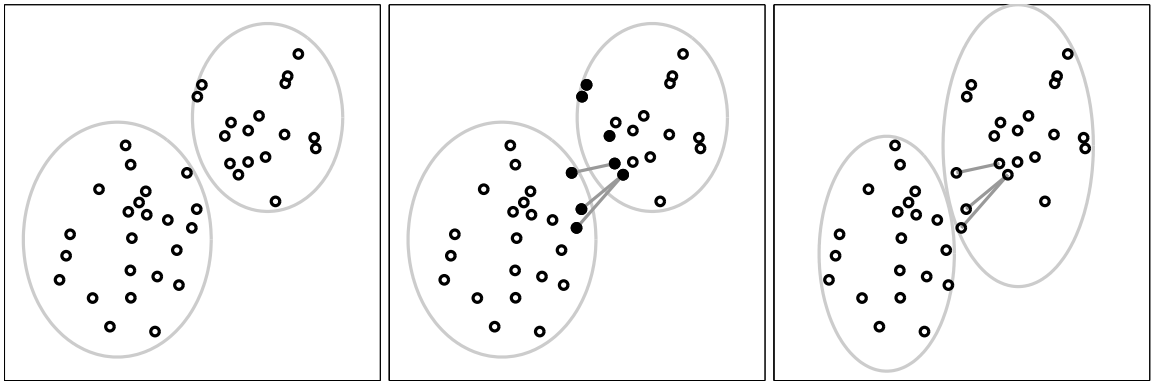
Entrées : Données $\mathbf{X}$ sous forme vectorielle, Q le nombre total de questions possibles, c le nombre de classes.
Sorties : Centres de gravité $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_c)$, ensemble des voisinages N.
début
  Choisir un premier objet $\mathbf{x}_1$ aléatoirement
  Placer $\mathbf{x}_1$ dans N_1 tel que $N = \{N_1\}$
  Soit $k \leftarrow 1$ le nombre de classes actuellement détectées
  tant que *(Le nombre Q de questions n'est pas dépassé et $k < c$)* **faire**
    Choisir un objet $\mathbf{x}_i$ éloigné de tous les points inclus dans N
    si $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$ *tel que* $\mathbf{x}_j \in N_l,\ \forall l \in \{1, \ldots, k\}$ **alors**
      $k \leftarrow k + 1$
      $N_k = \{\mathbf{x}_i\}$
    sinon
      Ajouter $\mathbf{x}_i$ au voisinage N_l pour lequel il est lié par une contrainte Must-Link.
  Calculer $\mathbf{V}$ (moyenne des points inclus dans N)
fin

Dans un deuxième temps et sous réserve que le quota de questions ne soit pas atteint, une phase de consolidation est mise en place. Cette dernière cherche à positionner au mieux les centres de gravité trouvés par la phase d'exploration. Ainsi, l'algorithme tente de découvrir en priorité des contraintes Must-Link (cf. algorithme 1.4).

La méthode AFCC : Cette méthode a été créée pour l'algorithme PCCA [31]. Dans un premier temps, l'algorithme est exécuté sans contrainte, ce qui permet d'obtenir une partition floue. À l'aide de ce résultat, les objets les moins certains, c'est-à-dire ceux qui se trouvent à la frontière des classes, sont identifiés. Pour cela, l'algorithme utilise une mesure d'hypervolume flou, qui permet d'évaluer la concentration des points dans une classe [28]. L'apprentissage actif consiste ensuite à associer deux objets incertains tels que ceux-ci ne sont pas dans la même classe mais sont proches l'un de l'autre. Après avoir choisi de cette manière Q paires d'objets, l'algorithme interroge l'expert sur le type de contraintes qui les lient. L'algorithme de classification PCCA est alors appliqué avec ces nouvelles contraintes (cf. figure 1.8). Il est ensuite possible de recommencer la procédure de sélection de contraintes et de refaire tourner l'algorithme.

Algorithme 1.4 : Pseudo-code de l'algorithme de consolidation de l'initialisation

Entrées : Données $\mathbf{X}$ sous forme vectorielle, centres de gravité $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_c)$, Q le nombre total de questions possibles.
Sorties : Centres de gravité $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_c)$.
début
  tant que *Le nombre Q de questions n'est pas dépassé* **faire**
    Prendre un objet $\mathbf{x}_i$ aléatoirement parmi les objets non contraints
    Trier les indices k des classes par ordre croissant des distances d_{ik}
    $k \leftarrow 1$
    tant que $(\mathbf{x}_i, \mathbf{x}_j) \notin \mathcal{M}$ **faire**
      $k \leftarrow k + 1$
    Recalculer $\mathbf{v}_k$ (moyenne des points inclus dans la classe k)
fin



(a) (b) (c)

Figure 1.8 – Schéma de l'apprentissage actif : exécution de PCCA sans contrainte pour obtenir une classification initiale (a), puis sélection des points incertains et choix des pairs contraints pour en demander le type à un expert (b), classification par PCCA avec les nouvelles contraintes (c), enfin retour en (b) si désiré.

La méthode de l'alpha-coupe : La méthode de l'alpha-coupe [38], basée sur les propriétés de la classification hiérarchique, est utilisée pour l'algorithme de classification par contraintes CCL (cf. paragraphe 1.2.1). La première étape consiste à exécuter CCL sans contrainte, ce qui revient à utiliser la classification ascendante hiérarchique avec comme critère de fusion le lien complet. En supposant qu'un expert peut répondre à Q contraintes, Klein & al proposent de remonter à la Q^{ieme} dernière fusion effectuée par CCL. L'algorithme demande alors à l'expert la contrainte qui lie deux objets inclus dans les deux sous-ensembles à fusionner. Dans le cas d'un Must-Link, les distances sont transformées à la manière de CCL et la fusion est effectuée. Dans le cas d'un Cannot-Link, seules les distances sont modifiées. De nouvelles fusions sont proposées à l'expert jusqu'à ce que le nombre Q de questions soit atteint.

Discussion

L'apprentissage actif, très étudié pour certains types d'algorithmes de classification semi-supervisée, commence tout juste à l'être dans le cas des algorithmes de

classification par contraintes. Comme les deux méthodes ont des comportements similaires, il est intéressant d'exploiter les observations effectuées dans l'une pour l'autre. Nous nous intéressons ainsi tout d'abord aux méthodes de classification avec ajout partiel de la connaissance des classes réelles des objets.

Dans ce cadre, certains auteurs soulignent l'existence de solutions contre-performantes en apprentissage actif. En effet il est parfois possible d'obtenir de moins bons résultats par cette stratégie que par le choix aléatoire d'objets à étiqueter [27, 32]. Cela peut également demander un plus grand nombre d'exemples étiquetés pour obtenir le même résultat qu'une sélection aléatoire [52]. Bien que ces résultats singuliers confirment le fait que l'apprentissage actif n'améliore pas forcément les résultats de classification, ils ne représentent qu'une partie des résultats possibles et sont souvent liés à des jeux de données particuliers.

Il arrive aussi que la stratégie d'apprentissage actif ne soit pas aboutie, car elle ne permet que l'exploitation d'une région particulière de l'espace. Ainsi, Dasgupta et Hsu montrent par l'exemple suivant que la sélection d'objets à étiqueter doit aussi suivre une notion d'exploration des données [13]. Pour cela, ils considèrent des données unidimensionnelles pour lesquelles la distribution est uniforme dans les quatre blocs présentés à la figure 1.9. Le jeu de données est divisé en deux classes. Les objets de la première classe se situent dans les blocs remplis d'un fond noir tandis que les objets de la seconde classe sont localisés dans les blocs de fonds blancs. Supposons qu'un algorithme de classification trouve sans connaissance a priori la frontière de décision f_1. Intuitivement, la stratégie d'apprentissage actif qui parait la plus appropriée correspond à celle qui sélectionne les exemples les plus proches de la frontière. Ces exemples sont en effet classés de manière moins certaine que les objets situés dans les blocs extrêmes. La frontière de décision converge alors vers f_2 et le taux d'erreur vers 5%. La frontière optimale f^* dont le taux d'erreur est à 2.5% n'est donc pas trouvée, car l'algorithme d'apprentissage actif n'explore pas d'autres régions que celles dont les exemples sont incertains.

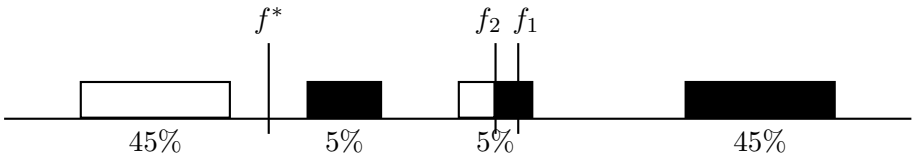


Figure 1.9 – Exemple de mauvaise stratégie d'apprentissage actif.

Ces contre-performances, aussi constatées en classification sous contraintes [63], permettent de se faire une idée des stratégies de sélection de contraintes à éviter. La recherche, encore novatrice dans le cas de contraintes Must-Link et Cannot-Link, peut aussi se tourner vers différentes études telles que la prise en compte de bruit dans les contraintes, la mise en place d'un critère d'arrêt plus évolué que le nombre maximal de questions, l'étude d'algorithmes d'exploration des données...

Enfin, notons qu'en classification par contraintes, et plus particulièrement quand des contraintes par paires sont utilisées, il est possible que la mise en place d'un apprentissage actif soit problématique. En effet, dans le cadre d'une application réelle où l'expert est sollicité, ce dernier peut avoir beaucoup de difficulté à trouver le lien existant entre deux objets. Si un objet est effectivement classé de manière incertaine par un algorithme de classification, il peut alors aussi être source de doute pour un expert. Cette dimension humaine doit donc être aussi prise en compte lors de la création d'une stratégie d'apprentissage actif.

Synthèse du chapitre

Le chapitre a permis d'introduire les différents concepts existant en classification. Un intérêt particulier a été porté aux méthodes de classification automatique et à leurs dérivés en classification sous contraintes. Ces modèles permettent d'intégrer des connaissances a priori afin d'améliorer le partitionnement des données. La forme la plus populaire de connaissances a priori, composée de contraintes de type Must-Link et Cannot-Link entre des paires d'objets, a été principalement étudiée. Enfin, une présentation de l'apprentissage actif dans le cadre de la classification semi-supervisée a été réalisée.

Classification automatique dans le cadre des fonctions de croyance

Dans ce chapitre, nous présentons des algorithmes de classification automatique qui ont été récemment développés dans un cadre évidentiel. Ces méthodes s'appuient sur la notion de partition crédale permettant la représentation de l'incertitude et de l'imprécision quant à l'appartenance des points aux classes. Dans un premier temps, le modèle mathématique sur lequel ces travaux sont fondés, nommé modèle des croyances transférables, est présenté. La description de deux algorithmes de classification évidentiels est réalisée dans une seconde partie.

2.1 Le modèle des croyances transférables

La théorie des probabilités a été longtemps considérée comme le seul cadre existant pour modéliser des connaissances imparfaites. Cette approche met l'accent sur l'incertitude qui existe sur les informations. À la fin du XIXe siècle, Georg Cantor propose d'utiliser des ensembles pour représenter l'imprécision des données. Plus tard, de nouveaux formalismes sont élaborés pour manipuler des données à la fois incertaines et imprécises. Ainsi, la théorie des possibilités [22], qui est issue de la théorie des sous-ensemble floues [72], décrit l'incertitude d'un événement par deux mesures de possibilité sur l'événement et son contraire. De même, la théorie des probabilités imprécises [65] considère un ensemble de probabilités permettant à l'incertitude d'être quantifiée de manière imprécise.

La théorie des fonctions de croyance (ou théorie de Dempster-Shafer) est fondée sur une étude effectuée par Dempster sur les bornes inférieure et supérieure d'une famille de distributions de probabilités [17]. Ce formalisme, développé ensuite par Shafer [54], permet de représenter l'incertain et l'imprécis par une notion de croyance à un événement. Considéré comme un ensemble de probabilités imprécisément connues, le modèle des fonctions de croyance généralise les modèles probabiliste et possibiliste. Smets et Kennes reprennent alors les travaux de Shafer pour proposer une interprétation subjectiviste de l'approche et en justifie axiomatiquement son existence [61]. Le modèle, appelé modèle des croyances transférables (MCT), définit les fonctions de croyance dans un cadre indépendant de tout modèle probabiliste. L'intérêt de ce cadre théorique a été démontré dans de nombreux domaines tels que le diagnostic [59], la reconnaissance de formes [20, 49, 62] et la fusion d'informations [29, 46, 60]. C'est ce modèle de représentation de données imparfaites qui est utilisé dans le mémoire.

L'une des spécificités du MCT est de considérer deux niveaux de raisonnement. Le niveau crédal permet la représentation et la manipulation des connaissances disponibles. Le niveau pignistique permet de prendre une décision sur la base de ces connaissances. Les principaux éléments mathématiques associées à ces deux niveaux sont développées dans cette partie.

2.1.1 Représentation de l'information

Concepts de base

Soient $\Omega = \{\omega_1, \ldots, \omega_c\}$ un ensemble fini d'éléments appelé cadre de discernement, et y une variable définie sur Ω. Les éléments $\{\omega_i\}_{i=1}^{c}$ sont considérés comme les hypothèses de l'état y d'un système. Dans le domaine de la classification par exemple, le cadre de discernement correspond aux différentes classes auxquelles un objet x peut être affecté, et y sa classe réelle. En théorie des fonctions de croyance, la connaissance partielle de la valeur de y est représentée par une fonction de masse de croyance :

Définition 2.1. *(Fonction de masse) Soit 2^Ω l'ensemble des parties de Ω. Une fonction de masse de croyance est une application $m : 2^\Omega \to [0, 1]$ telle que :*

$$\sum_{A \subseteq \Omega} m(A) = 1. \tag{2.1}$$

Chaque sous-ensemble $A \subseteq \Omega$ tel que $m(A) > 0$ est appelé élément focal de m. La valeur de l'élément focal $m(A)$ représente la part de croyance allouée à A qui, par manque d'information, ne peut pas être allouée à un sous-ensemble plus spécifique de A. Par conséquent, dans le cas où tous les éléments focaux sont des singletons, la fonction de croyance correspond à une distribution de probabilité.

Par définition, une fonction de masse est dite catégorique si un seul des sous-ensembles de Ω est un élément focal, c'est-à-dire s'il existe un unique $A \subseteq \Omega$ tel que $m(A) = 1$. Dans ce cas, l'ignorance totale est alors représentée par la fonction de masse vide ($m(\Omega) = 1$), et la certitude correspond au cas où A est un singleton.

Une fonction de croyance telle que $m(\emptyset) = 0$ est dite normale. Cette contrainte, utilisée généralement dans la théorie des fonctions de croyance, est relâchée dans le MCT. En effet, dans le MCT la quantité $m(\emptyset)$ s'interprète comme le degré de croyance que la valeur réelle de y ne soit pas inclus dans Ω [57]. En d'autres termes, le cadre de discernement est non exhaustif : c'est l'hypothèse d'un monde ouvert.

Il est parfois intéressant de passer de l'hypothèse d'un monde ouvert à l'hypothèse d'un monde fermé. Plusieurs méthodes de normalisation d'une fonction de masse ont été proposées. Ainsi, la méthode de Dempster [54] répartit le degré de conflit $m(\emptyset)$ entre tous les éléments focaux, alors que la normalisation de Yager [68] consiste à transférer $m(\emptyset)$ vers l'ensemble Ω.

Définition 2.2. *(Opération de normalisation) L'opération de normalisation qui permet d'obtenir une fonction de croyance normale m^* à partir d'une fonction de masse m non normalisée :*

– *Normalisation de Dempster :*

$$m^*(A) = \begin{cases} \frac{m(A)}{1-m(\emptyset)} & si \quad A \neq \emptyset, \\ 0 & sinon. \end{cases} \tag{2.2}$$

– *Normalisation de Yager :*

$$m^*(A) = \begin{cases} 0 & si \quad A = \emptyset, \\ m(\Omega) + m(\emptyset) & si \quad A = \Omega, \\ m(A) & sinon. \end{cases} \tag{2.3}$$

La connaissance exprimée par une fonction de masse peut également être représentée par les fonctions de croyance et de plausibilité. Ces trois fonctions correspondent à la même information sous différentes formes.

Définition 2.3. *(Fonctions de crédibilité et de plausibilité) Les fonctions de crédibilité bel et de plausibilité pl sont des applications* $bel : 2^\Omega \to [0,1]$ *et* $pl : 2^\Omega \to [0,1]$ *telles que :*

$$bel(A) = \sum_{B \subseteq A, B \neq \emptyset} m(B) \quad \forall A \subseteq \Omega, A \neq \emptyset, \quad et \quad bel(\emptyset) = 0, \tag{2.4}$$

$$pl(A) = \sum_{A \cap B \neq \emptyset} m(B) \quad \forall A \subseteq \Omega, A \neq \emptyset, \quad et \quad pl(\emptyset) = 0. \tag{2.5}$$

La quantité $bel(A)$ correspond au degré total de croyance spécifique et justifiée en A. En d'autres termes, seuls les sous-ensembles de A sont considérés, excepté l'hypothèse $\emptyset$ qui n'est pas retenue car celle-ci correspond à un sous-ensemble de A et de son contraire $\overline{A}$. La quantité $pl(A)$ représente le degré de croyance maximal qui peut potentiellement être attribué à l'hypothèse $y \in A$. Ces deux fonctions peuvent être vues comme les bornes inférieure et supérieure de la croyance attribuée à l'événement A (cf. figure 2.1). Comme elles représentent deux facettes de la même information, il est possible de retrouver l'une à partir de l'autre :

$$pl(A) = 1 - m(\emptyset) - bel(\overline{A}). \tag{2.6}$$

Exemple 2.1. *(L'énigme botanique) Un botaniste amateur cherche à constituer un herbier de la flore qui subsiste dans l'Oise. Il cueille une fougère du genre dryopteris dont il n'est pas sûr de reconnaître l'espèce. Il hésite entre trois plantes : l'affinis (exprimée par* p_1 *pour simplifier les notations), la gymnocarpium (notée* p_2*) et la filix-mas (notée* p_3*). Il constitue ainsi son cadre de discernement* Ω *avec ces trois espèces, et modélise sa connaissance par une fonction de masse* m_1 *représentée table 2.1. Notre botaniste a un degré de croyance spécifique de* 0.4 *que la fougère soit de l'espèce* p_1*, et un degré de croyance de* 0.3 *pour qu'elle soit du type* p_1 *ou* p_3*. Il a un degré d'ignorance totale de* 0.1 *quant à l'espèce de la fougère et une croyance de* 0.2 *que le spécimen cueilli corresponde à une toute autre plante. L'opération de normalisation utilisée est la méthode de Dempster-Shafer. Les fonctions de crédibilité* bel_1 *et de plausibilité* pl_1 *montrent les degrés de croyance minimale et maximale de chaque hypothèse.*

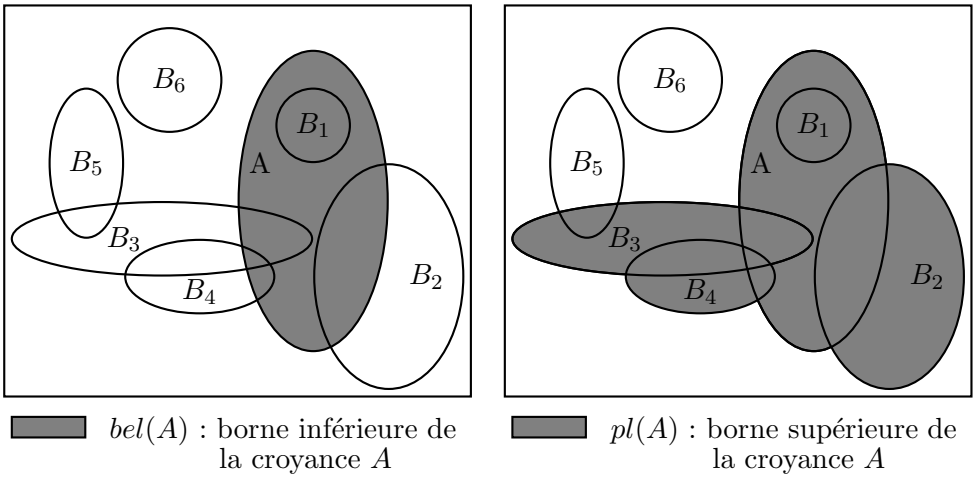


Figure 2.1 – Fonctions de crédibilité et de plausibilité.

A	$m_1(A)$	$m_1^*(A)$	$bel_1(A)$	$pl_1(A)$
$\emptyset$	0.2	0	0	0
$\{p_1\}$	0.4	0.5	0.4	0.8
$\{p_2\}$	0	0	0	0.1
$\{p_1, p_2\}$	0	0	0.4	0.8
$\{p_3\}$	0	0	0	0.4
$\{p_1, p_3\}$	0.3	0.375	0.7	0.8
$\{p_2, p_3\}$	0	0	0	0.3
$\Omega = \{p_1, p_2, p_3\}$	0.1	0.125	0.8	0.8

Tableau 2.1 – Fonctions de croyance pour une fougère de genre dryopteris

Travail sur un espace produit

Il est parfois intéressant de manipuler des fonctions de croyance définies sur des espaces produits, c'est-à-dire de raffiner ou de grossir un cadre de discernement. Les notions de marginalisation et d'extension sont particulièrement utiles dans ce contexte.

Définition 2.4. *(Marginalisation) Soit* $(\Omega \times \Theta)$ *un espace produit. La marginalisation de* $m^{\Omega\times\Theta}$ *sur* Ω *est la fonction de masse définie par :*

$$m^{(\Omega\times\Theta)\downarrow\Omega}(A) = \sum_{B\subseteq\Omega\times\Theta, B^{\downarrow\Omega}=A} m^{\Omega\times\Theta}(B) \qquad \forall A \subseteq \Omega, \tag{2.7}$$

avec $B^{\downarrow\Omega}$ la projection de B sur Ω telle que $B^{\downarrow\Omega} = \{\omega \in \Omega / \exists\theta \in \Theta,\ (\omega, \theta) \in B\}$.

Définition 2.5. *(Extension vide) Soit* $(\Omega \times \Theta)$ *un espace produit. L'extension vide de* m^{Ω} *sur l'espace produit* $\Omega \times \Theta$ *est la fonction définie par :*

$$m^{\Omega\uparrow(\Omega\times\Theta)}(B) = \begin{cases} m^{\Omega}(A) & si \quad B = A \times \Theta, \\ 0 & sinon. \end{cases} \qquad \forall B \subseteq \Omega \times \Theta \tag{2.8}$$

L'extension, contrairement à la marginalisation, permet d'agrandir le cadre de discernement. Il faut noter cependant que ces deux notions ne sont pas complémentaires. En effet, si la propriété $m^{\Omega\uparrow(\Omega\times\Theta)\downarrow\Omega} = m^{\Omega}$ est bien vérifiée, l'inverse ne l'est

pas toujours : $m^{(\Omega\times\Theta)\downarrow\Omega\uparrow(\Omega\times\Theta)} \neq m^{(\Omega\times\Theta)}$. Ceci est dû au fait que la marginalisation peut provoquer une perte d'information que par la suite l'extension n'est pas capable de recouvrer.

La marginalisation et l'extension sont des opérations utiles notamment pour combiner deux sources d'informations exprimées sur des cadres de discernement différents.

2.1.2 Combinaison d'informations

La combinaison d'informations permet d'agréger deux fonctions de croyance distinctes (c'est-à-dire deux sources d'informations indépendantes l'une de l'autre), pour obtenir une nouvelle fonction de masse plus informative. Il existe principalement deux opérations de combinaison d'informations qui ont la particularité d'être associatives et commutatives. La combinaison conjonctive, parfois dénommée règle de combinaison de Dempster non normalisée, est utilisée lorsque les deux sources sont supposées fiables. La combinaison disjonctive, plus prudente, permet de considérer le cas où il existe une source non fiable.

Définition 2.6. *(Combinaison conjonctive) Soient m_1 et m_2 deux fonctions de croyance distinctes. La combinaison conjonctive $(m_1 \textcircled{\cap} m_2)$ est la fonction de masse définie par :*

$$(m_1 \textcircled{\cap} m_2)(A) = \sum_{B\cap C=A} m_1(B)m_2(C) \quad \forall A \subseteq \Omega. \tag{2.9}$$

La quantité $(m_1 \textcircled{\cap} m_2)(\emptyset)$ représente le degré de conflit entre deux masses. Cette quantité peut être vue comme le degré de désaccord entre les deux sources d'information.

Définition 2.7. *(Combinaison disjonctive) Soient m_1 et m_2 deux fonctions de croyance distinctes. La combinaison disjonctive $(m_1 \textcircled{\cup} m_2)$ est la fonction de masse définie par :*

$$(m_1 \textcircled{\cup} m_2)(A) = \sum_{B\cup C=A} m_1(B)m_2(C) \quad \forall A \subseteq \Omega. \tag{2.10}$$

Exemple 2.2. *(L'énigme botanique, suite) Notre botaniste ne veut ajouter à son herbier que des plantes saines. Il lui faut donc vérifier l'état de la plante qu'il a cueillie. Pour cela, il définit le cadre de discernement $\Theta = \{s, \overline{s}\}$ tel que s correspond à un spécimen sain et $\overline{s}$ à un spécimen malade. Il détermine alors la fonction de masse suivante pour la plante qu'il a cueillie :*

A	$m_2(A)$
$\emptyset$	*0*
$\{s\}$	*0.8*
$\{\overline{s}\}$	*0.2*
$\{s, \overline{s}\}$	*0*

Le botaniste veut maintenant combiner les fonctions de masses définis sur les cadres de discernement différents Ω et Θ afin de prendre en compte toute les informations existantes dans une unique fonction de masses. Pour ce faire, il étend les deux fonctions de masses normalisées sur un espace commun $\Omega \times \Theta = \{(p_1, s), (p_2, s), (p_3, s), (p_1, \overline{s}), (p_2, \overline{s}), (p_3, \overline{s})\}$ à l'aide (2.8) :

A	$m_1^{\Omega\uparrow\Omega\times\Theta}(A)$
$\{(p_1, s), (p_1, \overline{s})\}$	*0.5*
$\{(p_1, s), (p_1, \overline{s}), (p_3, s), (p_3, \overline{s})\}$	*0.375*
$\{(p_1, s), (p_1, \overline{s}), (p_2, s), (p_2, \overline{s}), (p_3, s), (p_3, \overline{s})\}$	*0.125*

A	$m_2^{\Theta\uparrow\Omega\times\Theta}(A)$
$\{(\emptyset, s), (p_1, s), (p_2, s), (p_3, s)\}$	*0.8*
$\{(\emptyset, \overline{s}), (p_1, \overline{s}), (p_2, \overline{s}), (p_3, \overline{s})\}\}$	*0.2*

Il calcule ensuite la nouvelle fonction de masse $m_{12}^{\Omega\times\Theta}$ en appliquant la règle de combinaison conjonctive :

A	$m_{12}^{\Omega\times\Theta}(A)$
$\{(p_1, s)\}$	*0.4*
$\{(p_1, s), (p_3, s)\}$	*0.3*
$\{(p_1, s), (p_2, s), (p_3, s)\}$	*0.1*
$\{(p_1, \overline{s})\}$	*0.1*
$\{(p_1, \overline{s}), (p_3, \overline{s})\}$	*0.075*
$\{(p_1, \overline{s}), (p_2, \overline{s}), (p_3, \overline{s})\}$	*0.025*

2.1.3 Décision

Dans le cadre du MCT, le choix d'une hypothèse comprise dans le cadre de discernement Ω peut être réalisé principalement au niveau crédal. La décision peut ainsi être prise en sélectionnant l'hypothèse associée à une croyance maximale, ou à une plausibilité maximale. Cependant ces deux solutions aboutissent parfois à des décisions différentes. Pour résoudre cette singularité et obtenir un compromis entre les deux critères, il est possible d'utiliser la transformation pignistique qui permet convertir une fonction de masse normalisée en une distribution de probabilité [58]. Le choix d'une hypothèse peut ensuite être effectué par le biais de la théorie de la décision Bayésienne.

Définition 2.8. *(Transformation pignistique) Soit m^* une fonction de croyance normalisée. La transformation pignistique $BetP$ est une application $BetP : 2^\Omega - 1 \rightarrow [0, 1]$ telle que :*

$$BetP(\omega) = \sum_{\omega \in A} \frac{m^*(A)}{|A|}, \tag{2.11}$$

avec $|A|$ le cardinal de A.

La transformation pignistique répartit équitablement la masse de chaque sous-ensemble A entre les différents singletons inclus dans A.

Exemple 2.3. *(L'énigme botanique, suite et fin) Notre botaniste désire prendre une décision quant au genre de la fougère qu'il a cueilli et quant à son état de santé. La fonction de masse m_{12} étant déjà normalisée, il peut directement calculer la probabilité pignistique présentée table 2.2 et sélectionner l'événement de probabilité pignistique maximale. Ainsi, la fougère est probablement une dryopteris affinis en bonne santé.*

K	$BetP$
(p_1, s)	0.58
(p_2, s)	0.03
(p_3, s)	0.18
$(p_1, \overline{s})$	0.15
$(p_2, \overline{s})$	0.01
$(p_3, \overline{s})$	0.05

Tableau 2.2 – Probabilité pignistique pour une fougère de genre dryopteris.

2.1.4 Degré d'information d'une fonction de masse

Plusieurs mesures spécifiques aux fonctions de croyances ont été proposées. Elles permettent de quantifier l'incertitude d'une fonction de masse. La mesure la plus populaire est la mesure de non-spécificité qui permet d'évaluer le degré d'absence de connaissance [39]. Cette mesure est une généralisation de la mesure d'entropie proposée par [34].

Définition 2.9. *(Mesure de non-spécificité) La mesure de non-spécificité est une fonction définie telle que :*

$$N(m) = \sum_{A \subseteq \Omega \setminus \emptyset} m(A) \log_2 |A| + m(\emptyset) \log_2 |\Omega|, \tag{2.12}$$

avec $|A|$ le nombre d'éléments qui composent A.

Cette mesure, comprise entre 0 et $\log_2 |\Omega|$, est minimale lorsque les fonctions de croyances représentent une connaissance Bayésienne et maximale pour une fonction de masse vide ($m(\Omega) = 1$) ou pour une fonction de croyance telle que $m(\emptyset) = 1$.

Pour connaître la mesure de non-spécificité globale d'un ensemble de fonctions de croyances $m_1, \ldots, m_n$, il est possible d'utiliser la mesure suivante :

Définition 2.10. *(Mesure de non-spécificité globale) La mesure de non-spécificité globale et normalisée est une fonction définie telle que :*

$$N(\{m_1, \ldots m_n\}) = \frac{1}{n \log_2 |\Omega|} \sum_{i=1}^{n} N(m_i) \tag{2.13}$$

La mesure normalisée est comprise entre 0 et 1. Plus cette mesure s'approche de 1 et moins les fonctions de croyances sont spécifiques.

2.2 Algorithmes de classification évidentielle

Un des points forts de la théorie des fonctions de croyances est son aptitude à représenter des connaissances partielles. Cette particularité a d'abord été exploitée en discrimination [19, 20] puis en classification automatique [21, 44, 45]. En classification automatique, le MCT permet de trouver à partir des données un partitionnement très riche en information. En effet, la partition renvoyée, nommée partition

crédale, est construite à partir de fonctions de croyance et permet d'exprimer de manière naturelle le doute sur l'affectation d'objets à une classe particulière.

Dans un premier temps nous définissons le concept de partition crédale. Deux algorithmes de classification automatique de la littérature utilisant les fonctions de croyance sont ensuite présentés. Ils serviront de base au travail qui sera développé dans les chapitres 3 et 4.

2.2.1 La notion de partition crédale

Dans le cadre d'une classification automatique, $O = \{o_1, \dots o_n\}$ définit un ensemble de n objets à classifier et $\Omega = \{\omega_1, \dots, \omega_c\}$ les c classes dans lesquelles les objets peuvent être affectés. La connaissance partielle de l'appartenance d'un objet o_i peut alors être représentée par une fonction de masse m_i. De cette manière, un degré de croyance peut être attribué non pas uniquement aux singletons, mais à n'importe quel sous-ensemble de Ω. Cette représentation permet donc de modéliser de nombreuses situations allant de l'ignorance totale à la certitude complète (cf. exemple 2.4). La matrice $\mathbf{M}$, nommée partition crédale, est constituée des masses de croyance pour chaque individu.

Exemple 2.4. *(La partition crédale) La table 2.3 présente une partition crédale de cinq objets devant être classés dans deux classes. Les classes du premier et du deuxième objet sont connues avec certitude alors que la classe du quatrième objet est totalement inconnue. Le troisième objet correspond à une connaissance Bayésienne du problème. Le cinquième objet, considéré comme un objet atypique car il n'appartient à aucune des classes présente dans Ω, est caractérisé par $m(\emptyset) = 1$. Cet objet traduit souvent un objet bruité, c'est-à-dire un objet dont les caractéristiques contiennent des erreurs (par exemple des erreurs de mesures).*

A	$m_1(A)$	$m_2(A)$	$m_3(A)$	$m_4(A)$	$m_5(A)$
$\emptyset$	0	0	0	0	1
ω_1	1	0	0.8	0	0
ω_2	0	1	0.2	0	0
Ω	0	0	0	1	0

Tableau 2.3 – Exemple de partition crédale.

La partition crédale $\mathbf{M}$ peut donc être vue comme un modèle général de partitionnement :

- Si pour chaque objet o_i la fonction de masse m_i est certaine, c'est-à-dire que la croyance est allouée de manière totale à un singleton, alors $\mathbf{M}$ définit une partition dure (appelée aussi partition nette). Cette situation correspond à une certitude complète du partitionnement.
- Si pour chaque objet o_i la fonction de masse m_i représente une connaissance Bayésienne, alors $\mathbf{M}$ exprime une partition floue.

Par la suite et afin de simplifier les notations, le degré de croyance $m_i(A_j)$ que l'objet o_i appartienne au sous-ensemble $A_j \subseteq \Omega$ sera noté m_{ij}.

Comme le soulignent Masson et Denœux [44], une partition crédale est une représentation des données riche qui peut être transformée de diverses manières afin d'aider l'utilisateur à interpréter les résultats. L'opération la plus courante consiste à convertir une partition crédale en une partition floue à l'aide de la transformation pignistique (cf. équation (2.11)). Cette partition, plus classique, est souvent mieux connue des experts et donc plus simple d'interprétation pour eux. Une autre manière intéressante de synthétiser les informations consiste à affecter chaque objet au sous-ensemble de plus forte masse. Ainsi, au maximum 2^c groupes peuvent être obtenus. Cette opération fournit une partition nommée partition crédale dure. Elle permet de détecter d'une part les objets qui sont attribués sans ambiguïté à une classe et d'autre part les objets proches d'une frontière. Enfin, la partition crédale dure peut être exploitée afin de représenter les estimations haute et basse des classes. Une approximation haute correspond, pour chaque classe k, à l'ensemble des points qu'is serait plausible d'affecter à ω_k. Cet ensemble de points est défini comme l'union des sous-ensembles de la partition crédale dure qui font intervenir la classe ω_k. À l'inverse, l'approximation basse permet de détecter les objets classés sans ambiguïté. Elle correspond donc à la partition crédale dure privée des éléments focaux non singletons.

2.2.2 ECM

La version crédibiliste de l'algorithme des c-moyennes, nommée ECM [44], a pour objectif de calculer une partition crédale à partir d'un tableau individus-variables.

Expression de la fonction objectif

Déterminer une partition crédale revient à calculer, pour chaque objet o_i, la quantité m_{ij} $\forall A_j \neq \emptyset, A_j \subseteq \Omega$ en fonction de la distance d_{ij} entre o_i et le centre de gravité du sous-ensemble A_j.

Comme dans FCM, chaque classe est caractérisée par un centre de gravité $\mathbf{v}_k \in \mathbb{R}^p$. Pour tous les sous-ensembles $A_j \subseteq \Omega, A_j \neq \emptyset$ non singletons, un centre de gravité $\overline{\mathbf{v}}_j$ est défini tel qu'il représente le barycentre des centres associés aux classes de A_j :

$$\overline{\mathbf{v}}_j = \frac{1}{|A_j|} \sum_{l=1}^{c} s_{lj} \mathbf{v}_l, \quad \text{avec} \quad s_{lj} = \begin{cases} 1 & \text{si } \omega_l \in A_j, \\ 0 & \text{sinon.} \end{cases} \tag{2.14}$$

L'ensemble des barycentres est alors noté $\mathbf{V}$. La distance d_{ij} entre l'objet o_i et le sous-ensemble A_j correspond, dans le cas d'une distance Euclidienne, à l'équation suivante :

$$d_{ij} = ||\mathbf{x}_i - \overline{\mathbf{v}}_j||. \tag{2.15}$$

L'algorithme ECM recherche les matrices $\mathbf{M}$ et $\mathbf{V}$ qui minimise un critère similaire à celui de l'algorithme NC (cf. équation (1.11)) :

$$J_{ECM}(\mathbf{M}, \mathbf{V}) = \sum_{i=1}^{n} \sum_{A_j \neq \emptyset} |A_j|^{\alpha} m_{ij}^{\beta} d_{ij}^2 + \sum_{i=1}^{n} \rho^2 m_{i\emptyset}^{\beta}, \tag{2.16}$$

sous les contraintes

$$\begin{cases} m_{ij} \geq 0 & \forall i = \{1, \dots n\}, \quad \forall j / A_j \subseteq \Omega \\ \sum\limits_{j/A_j \subseteq \Omega, A_j \neq \emptyset} m_{ik} + m_{i\emptyset} = 1 & \forall i = \{1, \dots n\}. \end{cases} \tag{2.17}$$

où $m_{i\emptyset}$ représente la quantité de croyance allouée à l'ensemble vide pour l'élément i. Ce sous-ensemble $\emptyset$ est considéré comme une "classe de bruit" qui permet de détecter les objets atypiques ; il est donc traité séparément des autres sous-ensembles. Le paramètre ρ désigne une distance fixe à l'ensemble des classes, au-delà de laquelle un objet est considéré comme atypique. Le coefficient $|A_k|^\alpha$ est introduit de manière à pénaliser l'allocation de croyance aux sous-ensembles de forte cardinalité. L'exposant α permet le contrôle de cette pénalisation.

De la même manière que FCM ou NC, la partition crédale est calculée en minimisant la fonction objectif de manière alternée par rapport aux masses et aux centres de gravité. Les conditions nécessaires d'optimalité pour les masses donnent des équations de mise à jour directe. Ces équations sont très similaires à celles de NC, excepté qu'il existe 2^c valeurs de m_{ij} pour ECM, et $c+1$ degrés d'appartenance u_{ik} pour NC. Une règle de mise à jour plus complexe est trouvée pour les isobarycentres : les conditions d'optimalité équivalent à résoudre un système d'équations linéaires. L'annexe B détaille ces différents calculs. L'algorithme ECM, tout comme les c-moyennes et ses dérivés, commence par une initialisation de la partition crédale et optimise alternativement $\mathbf{V}$ et $\mathbf{M}$ jusqu'à convergence de la solution.

Choix des hyper-paramètres

L'algorithme comprend un certain nombre de paramètres à fixer avant son exécution. Masson et Denœux [44] proposent plusieurs pistes de réflexions pour aider l'utilisateur quant aux choix de ces paramètres.

- Le coefficient β, fixé à une forte valeur, contribue de la même manière que pour les algorithmes FCM et NC à durcir la partition, c'est-à-dire à favoriser pour chaque objet l'allocation totale de croyance à un sous-ensemble de Ω. Le choix par défaut de β pour tous les algorithmes l'utilisant consiste en $\beta = 2$.
- Le coefficient α contrôle la quantité de croyance attribuée aux sous-ensembles de Ω ayant une forte cardinalité. Plus la valeur de α sera élevée et plus la partition crédale tend vers une partition floue. Le choix de cette valeur dépend donc essentiellement de ce qu'un utilisateur désire obtenir. Par défaut, α est fixé à 1.
- Le coefficient ρ fixe une distance entre les objets et la classe de bruit. Sa valeur peut varier de manière importante selon les données. Tout comme pour l'algorithme NC (cf. partie 1.1.2, Algorithme des c-moyennes floues et gestion de données bruitées), ce paramètre peut être modifié afin de devenir moins dépendant des données :

$$\rho^2 = \lambda \frac{1}{cn} \left(\sum_{i=1}^{n} \sum_{j=1}^{c} d_{ij}^2 \right), \tag{2.18}$$

avec λ un taux de rejet. Les distances correspondent à une première expérience sans ensemble vide. Si un expert pense que les données ne sont pas bruitées, alors λ doit être fixé à une forte valeur.

Afin de limiter la complexité de l'algorithme, il peut être intéressant de choisir des sous-ensembles particuliers de Ω. En effet, le nombre de classes c fait varier de façon exponentielle le nombre de sous-ensembles associés aux classes. Par conséquent, les temps de calculs augmentent eux aussi de façon exponentielle par rapport à ce nombre de classes. Au-delà d'une dizaine de classes, les auteurs proposent donc de limiter l'ensemble des éléments focaux aux singletons, à l'ensemble vide et à Ω.

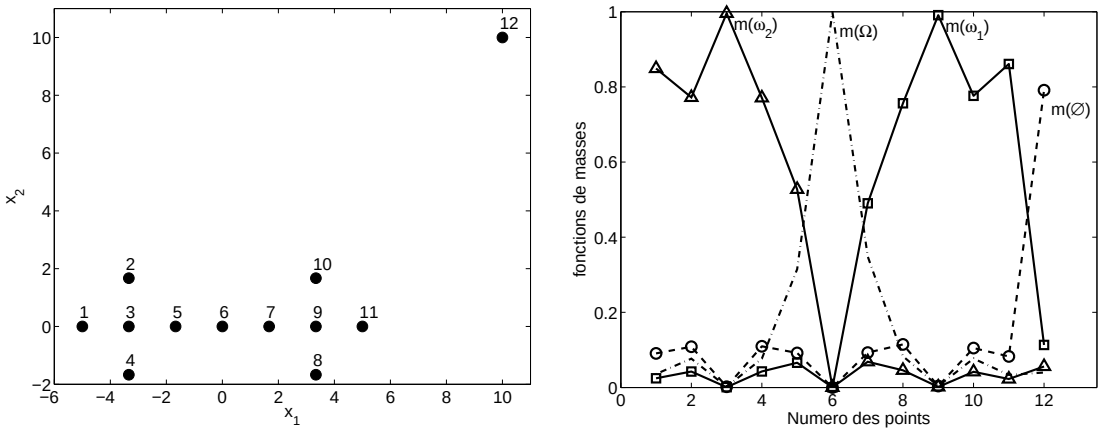
Enfin, remarquons que le problème fondamental du choix du nombre de classes peut être résolu par le calcul d'un indice de validité. Cet indice est calculé pour différentes valeurs de c. La valeur optimale de c est alors déterminée en cherchant un minimum, un maximum, ou un changement brusque du critère.

Pour l'algorithme ECM, il est possible d'observer que si le nombre de classes est correct, les centres de gravité se trouvent dans des zones de forte densité et par conséquent les masses sont distribuées en majorité entre les singletons. À l'inverse, la méthode compense un nombre de classes trop petit ou trop grand par l'affectation d'une partie de la masse aux sous-ensembles de fortes cardinalité et à l'ensemble vide. En d'autres termes, plus la partition crédale est spécifique, plus la structure inhérente aux données trouvée est cohérente. Les auteurs proposent donc d'utiliser la mesure de non-spécificité globale $N(\mathbf{M})$ (cf. équation (2.10)) comme indice de validité. Plus $N(\mathbf{M})$ est petit et plus la partition crédale est spécifique. L'indice doit donc être minimisé. Il faut noter que $N(\mathbf{M})$ est fortement lié au paramètre α, qui doit être fixé à une valeur suffisamment petite pour obtenir une partition non floue. Dans le cas contraire, un indice de validité propre aux partitions floues doit être utilisé.

Exemples illustratifs

Afin d'illustrer le comportement de l'algorithme de classification ECM, deux exemples sont présentés. Le premier considère un jeu de données bidimensionnelles nommé diamant, inspiré d'un jeu classique [66, 44]. Comme le montre la figure 2.2(a), il est composé de 12 objets et le dernier correspond à un individu bruité. La partition crédale, trouvée par ECM avec les paramètres $c = 2$, $\alpha = 1$, $\beta = 2$ et $\rho^2 = 20$, est représentée par la figure 2.2(b). Les masses montrent que les deux classes sont correctement trouvées. L'objet 6, caractérisé par une masse de croyance élevée allouée à Ω, se révèle être un point ambigu qui peut être affecté aussi bien à la classe ω_1 qu'à ω_2. Le point 12 quant à lui est correctement détecté comme étant un objet atypique : une grande partie de sa masse est allouée à l'ensemble vide.

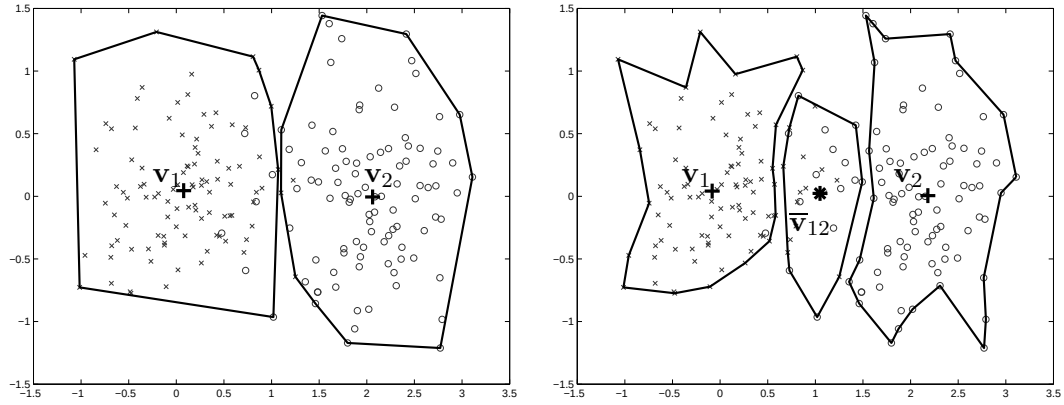
La figure 2.3 illustre le concept de barycentre et la différence entre les algorithmes FCM et ECM. Un jeu de données en 2D est généré à partir de deux gaussiennes et doit être séparé en deux classes. Tout comme FCM, ECM génère deux centres de gravités v_1 et v_2 correspondant aux deux classes. Il calcule cependant aussi un isobarycentre v_{12} associé à Ω en moyennant les centres v_1 et v_2. Les points incertains,



(a) (b)

Figure 2.2 – Jeu de données diamant (a) et sa partition crédale obtenue avec ECM (b).

caractérisés par une masse élevée allouée à Ω, se situent à la frontière entre les deux classes.



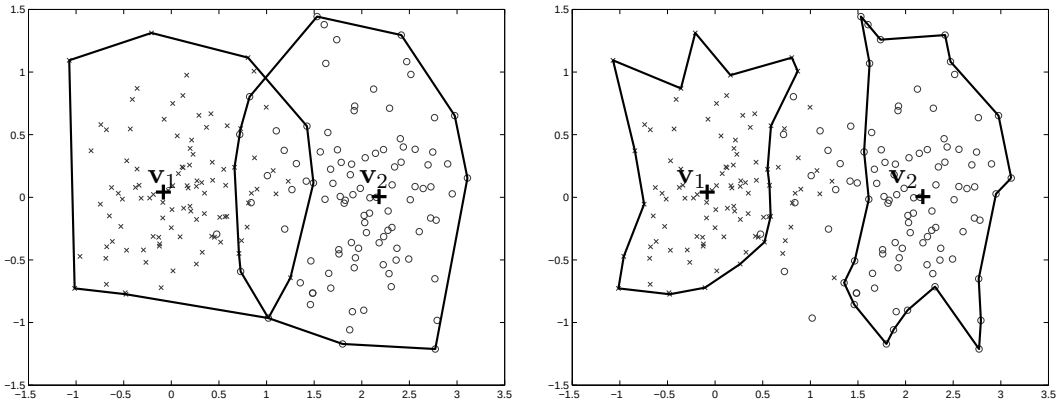
(a) (b)

Figure 2.3 – Centres de gravité et partitions obtenues avec le maximum de probabilité pour FCM (a) et le maximum de fonction de croyance pour ECM (b), avec $\alpha = 1$, $\beta = 2$ et $\rho^2 = 20$. Les signes représentent les classes réelles des objets et les lignes correspondent aux frontières des classes pour (a) et des sous-ensembles pour (b). Les centres de gravité des classes $\mathbf{v}_1$ et $\mathbf{v}_2$ sont indiqués par des croix tandis que le centre de gravité de Ω, nommé $\mathbf{v}_{12}$, est représenté par une étoile.

Les estimations haute et basse des classes pour ce jeu de données sont montrées à la figure 2.4. Ces estimations sont parfois vues comme des représentations plus aisée à interpréter que la représentation d'une partition crédale nette.

2.2.3 EVCLUS

L'algorithme ECM, qui est une extension directe de l'algorithme NC, ne peut être utilisé que dans le cas de données vectorielles. La méthode évidentielle nommée EVCLUS, permet quant à elle la manipulation de données relationnelles. Le principe de l'algorithme EVCLUS consiste à construire une partition crédale à partir



(a) (b)

Figure 2.4 – Estimations haute (a) et basse (b) obtenues à l'aide de la partition crédale nette de l'algorithme ECM avec $\alpha = 1$, $\beta = 2$ et $\rho^2 = 20$ sur le jeu de données en 2D généré à partir de deux Gaussiennes. Les signes représentent les classes réelles des objets et les lignes correspondent aux frontières haute et basse des classes. Les centres de gravité des classes $\mathbf{v}_1$ et $\mathbf{v}_2$ sont indiqués par des croix.

d'une matrice de dissimilarité. Pour cela, Denœux et Masson proposent un modèle [21] qui s'inspire des méthodes de positionnement multidimensionnel (en anglais, Multidimensional Scaling ou MDS [9]).

Expression de la fonction objectif

Intuitivement, il est possible de dire que plus deux objets sont proches et plus il est plausible qu'ils appartiennent à la même classe. Pour formaliser ce concept, les auteurs proposent d'exprimer la plausibilité que deux objets soient dans la même classe. Soient o_i et o_j deux objets décrits par deux fonctions de masses m_i et m_j. En se plaçant dans $\Omega^2 = \Omega \times \Omega$ il est possible de calculer une fonction de masse quantifiant la croyance sur l'appartenance conjointe des individus o_i et o_j. Cette fonction de masse, notée $m_{i\times j}$, s'obtient en étendant m_i et m_j sur Ω^2 à l'aide de l'équation (2.8), puis en les combinant par la règle conjonctive (cf. équation (2.9)). Le résultat de cette combinaison est :

$$m_{i\times j}(A \times B) = m_i(A)m_j(B) \quad \forall A, B \subseteq \Omega, A \neq \emptyset, B \neq \emptyset, \tag{2.19}$$

$$m_{i\times j}(\emptyset) = m_i(\emptyset) + m_j(\emptyset) - m_i(\emptyset)m_j(\emptyset). \tag{2.20}$$

À partir de $m_{i\times j}$, il est possible de calculer la plausibilité que les objets o_i et o_j appartiennent ou non à la même classe. Soit $\theta \in \Omega^2$ l'événement "o_i et o_j sont dans la même classe" $\theta = \{(\omega_1, \omega_1), \ldots, (\omega_c, \omega_c)\}$. Soit $pl_{i\times j}$ la fonction de plausibilité associée à $m_{i\times j}$. La quantité $pl_{i\times j}(\theta)$ peut donc être calculée :

$$pl_{i\times j}(\theta) = \sum_{A\cap B\neq\emptyset} m_i(A)m_j(B), \tag{2.21}$$

$$= 1 - \sum_{A\cap B=\emptyset} m_i(A)m_j(B), \tag{2.22}$$

$$= 1 - K_{ij}, \tag{2.23}$$

avec K_{ij} le degré de conflit entre m_i et m_j.

Le problème revient alors à trouver la partition crédale $\mathbf{M}$ telle que deux objets similaires sont caractérisés par des masses de croyance avec un faible degré de conflit et deux objets éloignés sont associés à des masses conflictuelles. L'algorithme EVCLUS cherche ainsi à minimiser l'écart entre la matrice $\mathbf{K} = (K_{ij})$ de taille $(n \times n)$ et la matrice de dissimilarité $\mathbf{D}$. Cette minimisation passe par l'optimisation d'une fonction objectif similaire à la fonction de stress de Sammon normalisée [51] :

$$J_{EVCLUS}(\mathbf{K}, a, b) = \frac{1}{C} \sum_{i<j} \frac{(aK_{ij} + b - d_{ij})^2}{d_{ij}}, \tag{2.24}$$

avec C un coefficient de normalisation tel que :

$$C = \sum_{i<j} d_{ij}, \tag{2.25}$$

sous les contraintes (2.17). Les coefficients a et b permettent de modifier les bornes du conflit K_{ij} (compris entre 0 et 1) afin de s'ajuster aux valeurs de la matrice de dissimilarité. Le dénominateur d_{ij} permet d'accorder un poids plus important aux faibles dissimilarités.

Afin d'éviter les problèmes d'optimisation liés aux contraintes (2.17), les auteurs proposent de les supprimer en utilisant le changement de variables suivant :

$$m_i(A_k) = \frac{\exp(\alpha_{ik})}{\sum_{l=1}^{2^c} \exp(\alpha_{il})}. \tag{2.26}$$

Par la suite, $\mathbf{M}$ correspond à l'ensemble des α_{ik} $\forall i \in \{1, \ldots, n\}, k/A_k \subseteq \Omega$. La fonction objectif est minimisée itérativement en fonction de $\mathbf{M}$, a et b. Ces paramètres sont optimisés à l'aide d'une méthode de descente de gradient décrit à l'annexe C. Cette procédure d'optimisation est dépendante de l'initialisation : la partition finale trouvée peut correspondre à un minimum local. Par conséquent, l'algorithme EVCLUS est généralement initialisé plusieurs fois et la solution retenue est celle pour laquelle J_{EVCLUS} est minimal.

Choix des hyper-paramètres

Les hyper-paramètres correspondent ici aux paramètres indissociables d'un algorithme de classification automatique de type évidentiel. Ces paramètres sont le nombre de classes c et le choix des sous-ensembles A_k.

Comme expliqué dans la partie consacrée à ECM, la plupart des méthodes qui permettent de choisir le nombre de classes sont basées sur un critère de validité calculé pour différentes valeurs de c. Un minimum, un maximum, ou un changement brusque de la courbe construite indique le nombre de classes à retenir. L'indice de validité utilisé pour EVCLUS dans [21] est la valeur du stress J_{EVCLUS}, mais cela pourrait tout aussi bien être l'indice employé dans ECM.

Tout comme ECM, l'algorithme EVCLUS ne peut pas fonctionner avec un nombre important de classes. Sa technique d'optimisation est en effet itérative et limite l'algorithme à moins de 10 classes. Par conséquent, la sélection de sous-ensembles particuliers par un expert permet de réduire le temps d'exécution de l'algorithme et de dépasser cette limitation.

Exemple illustratif

Afin d'illustrer le comportement de l'algorithme EVCLUS, le jeu diamant présenté figure 2.2(a) est utilisé. Une matrice des distances Euclidienne entre chaque objet est calculée (cf. table 2.4) et un treizième point est ajouté (numéro 1) de manière à ce que celui-ci soit proche de tous les objets à l'exception de celui bruité. Cet objet, qui n'a aucune représentation graphique, est un point atypique qui peut parfois se rencontrer dans certains jeux de données relationnelles.

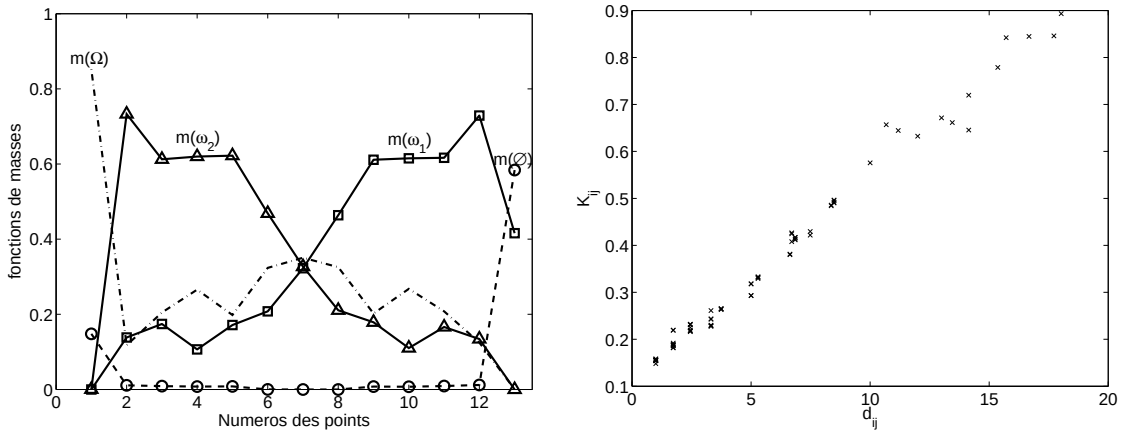
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	1	1	1	1	1	1	1	1	200
2	1	0	6	3	6	11	25	44	72	70	72	100	325
3	1	6	0	3	11	6	14	28	56	47	45	72	247
4	1	3	3	0	3	3	11	25	47	45	47	70	278
5	1	6	11	3	0	6	14	28	45	47	56	72	314
6	1	11	6	3	6	0	3	11	28	25	28	44	236
7	1	25	14	11	14	3	0	3	14	11	14	25	200
8	1	44	28	25	28	11	3	0	6	3	6	11	169
9	1	72	56	47	45	28	14	6	0	3	11	6	181
10	1	70	47	45	47	25	11	3	3	0	3	3	144
11	1	72	45	47	56	28	14	6	11	3	0	6	114
12	1	100	72	70	72	44	25	11	6	3	6	0	125
13	200	325	247	278	314	236	200	169	181	144	114	125	0

Tableau 2.4 – Matrice de dissimilarité du jeu de données Diamant.

Après une exécution de EVCLUS, nous remarquons qu'à l'instar de ECM, le point bruité se distingue par une forte masse affectée à l'ensemble vide et l'objet 7, au centre des deux classes, présente une masse importante sur l'ensemble Ω (cf. figure 2.5(a)). L'individu 1, proche de tous les points, est correctement affecté à l'ensemble Ω. La figure 2.5(b) présente les degrés de conflit K_{ij} par rapport aux distances D_{ij} (cette figure est parfois appelée diagramme de Shepard dans la littérature). Elle permet de visualiser la relation entre les deux variables : nous remarquons effectivement que plus la distance est grande et plus le conflit est important.

Synthèse du chapitre

Dans ce chapitre, nous avons vu les principaux concepts du MCT. Ce modèle permet de représenter des informations incertaines et imprécises. Dans le domaine de la classification automatique, il permet de définir la notion de partition crédale. Cette dernière permet de décrire de nombreuses situations allant de l'ignorance totale à la



(a) (b)

Figure 2.5 – Partition crédale (a) et diagramme de Shepard (b) obtenue avec EVCLUS pour le jeu de données diamant.

certitude parfaite quant à l'affectation d'un objet à une classe. Nous présentons deux algorithmes de classification automatique qui génèrent une partition crédale. L'un d'eux admet en entrée des données individus-variables alors que le second traite une matrice de dissimilarité. Ces deux algorithmes ont servi de base au travail présenté dans ce mémoire.

ECM avec contraintes

Dans ce chapitre, nous introduisons deux modifications dans l'algorithme de classification évidentielle ECM afin d'améliorer ses performances. Dans un premier temps, nous modifions l'algorithme pour permettre la détection de classes ellipsoïdales. Cette modification va faciliter par la suite la prise en compte de contraintes de types Must-Link et Cannot-Link. Dans un deuxième temps, nous montrons comment intégrer ces contraintes dans ECM. La nouvelle méthode, baptisée CECM (Constrained Evidential C-Means), est ensuite évaluée sur plusieurs jeux de données.

3.1 ECM avec une métrique adaptative

3.1.1 Expression de la métrique

La version de base de ECM utilise la distance Euclidienne. Les classes sont donc supposées sphériques. Cependant, l'utilisation de la distance de Mahalanobis peut être intéressante dans le cas où les classes ont une forme ellipsoïdale. De la même manière que Gustafson et Kessel (cf. partie 1.1.2, Algorithme des c-moyennes floues et métrique adaptative), nous associons à chaque classe ω_k une matrice $\mathbf{S}_k$ permettant d'obtenir des classes de forme ellipsoïdale, d'orientation et de taille spécifiques. Tout comme dans le calcul des centres pour ECM, la matrice $\overline{\mathbf{S}}_j$ associée à un sous-ensemble $A_j \subseteq \Omega$ correspond à la moyenne pondérée des matrices associées aux classes composant A_j :

$$\overline{\mathbf{S}}_j = \frac{1}{|A_j|} \sum_{l=1}^{c} s_{lj} \mathbf{S}_l \quad \text{avec} \quad s_{lj} = \begin{cases} 1 \text{ si } \omega_l \in A_j, \\ 0 \text{ sinon.} \end{cases} \tag{3.1}$$

Le calcul de la distance correspond alors, pour chaque chaque sous-ensemble $A_j \neq \emptyset$, à l'équation suivante :

$$d_{ij}^2 = (\mathbf{x}_i - \overline{\mathbf{v}}_j)^\top \overline{\mathbf{S}}_j (\mathbf{x}_i - \overline{\mathbf{v}}_j). \tag{3.2}$$

L'ensemble des matrices $\mathbf{S} = \{\mathbf{S}_1, \ldots, \mathbf{S}_c\}$ représente alors un nouveau paramètre à optimiser dans la fonction objectif $J_{ECM}(\mathbf{M}, \mathbf{V}, \mathbf{S})$. Cependant, un minimum global de J_{ECM} est atteint si toutes les matrices $\mathbf{S}_j$ sont nulles. Par conséquent, à l'instar de Gustafson et Kessel [33], nous ajoutons une contrainte de volume sur les ellipses de chaque classe :

$$\det(\mathbf{S}_j) = \varrho_j, \tag{3.3}$$

où les paramètres $\varrho_j > 0$ sont fixés a priori.

3.1.2 Optimisation

La minimisation de la fonction objectif est réalisée en alternant le calcul de la partition crédale $\mathbf{M}$, des centres de gravité $\mathbf{V}$ et de la métrique $\mathbf{S}$.

Mise à jour de la partition crédale M

La minimisation de la fonction objectif J_{ECM} par rapport à $\mathbf{M}$ est indépendante de la métrique. Ainsi les équations de mise à jour des masses pour ECM avec une métrique adaptative sont identiques à celle d'ECM avec une distance Euclidienne :

$$m_{ij} = \frac{|A_j|^{-\alpha/(\beta-1)} d_{ij}^{-2/(\beta-1)}}{\sum_{A_k \neq \emptyset} |A_k|^{-\alpha/(\beta-1)} d_{ik}^{-2/(\beta-1)} + \rho^{-2/(\beta-1)}} \qquad \forall i \in \{1,\dots,n\},\ A_j \neq \emptyset, \quad (3.4)$$

et

$$m_{i\emptyset} = 1 - \sum_{A_j \neq \emptyset} m_{ij} \qquad \forall i \in \{1,\dots,n\}. \quad (3.5)$$

Mise à jour des centres de gravité

Les matrices $\mathbf{M}$ et $\mathbf{S}$ sont maintenant supposées fixes. La minimisation de J_{ECM} par rapport à $\mathbf{V}$ est un problème d'optimisation sans contraintes. Les dérivées partielles correspondantes sont :

$$\frac{\partial J_{ECM}}{\partial \mathbf{v}_l} = \sum_{i=1}^{n} \sum_{A_j \neq \emptyset} |A_j|^{\alpha} m_{ij}^{\beta} \frac{\partial d_{ij}^2}{\partial \mathbf{v}_l} \qquad \forall\ l \in \{1,\dots,c\}, \quad (3.6)$$

$$\frac{\partial d_{ij}^2}{\partial \mathbf{v}_l} = 2 s_{lj} \overline{\mathbf{S}}_j (\mathbf{x}_i - \overline{\mathbf{v}}_j) \left(-\frac{1}{|A_j|} \right) \qquad \forall\ l \in \{1,\dots,c\}. \quad (3.7)$$

En utilisant (3.6), (3.7) et l'expression (2.14) de $\overline{\mathbf{v}}_j$, nous obtenons, pour tout $l \in \{1,\dots,c\}$:

$$\begin{aligned} \frac{\partial J_{ECM}}{\partial \mathbf{v}_l} &= -2 \sum_{i=1}^{n} \sum_{A_j \neq \emptyset} |A_j|^{\alpha-1} m_{ij}^{\beta} s_{lj} \overline{\mathbf{S}}_j (\mathbf{x}_i - \overline{\mathbf{v}}_j), & (3.8) \\ &= -2 \sum_{i=1}^{n} \sum_{A_j \neq \emptyset} |A_j|^{\alpha-1} m_{ij}^{\beta} s_{lj} \overline{\mathbf{S}}_j \left(\mathbf{x}_i - \frac{1}{|A_j|} \sum_{k=1}^{c} s_{kj} \mathbf{v}_k \right). & (3.9) \end{aligned}$$

En annulant ces dérivées partielles, nous obtenons les c équations suivantes :

$$\begin{aligned} \frac{\partial J_{ECM}}{\partial \mathbf{v}_l} = 0 \Rightarrow & \sum_{i=1}^{n} \sum_{A_j \neq \emptyset} |A_j|^{\alpha-1} m_{ij}^{\beta} s_{lj} \overline{\mathbf{S}}_j \mathbf{x}_i = \sum_{k=1}^{c} \sum_{i=1}^{n} \sum_{A_j \neq \emptyset} |A_j|^{\alpha-2} m_{ij}^{\beta} s_{lj} s_{kj} \overline{\mathbf{S}}_j \mathbf{v}_k, & (3.10) \\ \Rightarrow & \sum_{i=1}^{n} \sum_{A_j \ni \omega_l} |A_j|^{\alpha-1} m_{ij}^{\beta} \overline{\mathbf{S}}_j \mathbf{x}_i = \sum_{k=1}^{c} \sum_{i=1}^{n} \sum_{A_j \supseteq \{\omega_k, \omega_l\}} |A_j|^{\alpha-2} m_{ij}^{\beta} \overline{\mathbf{S}}_j \mathbf{v}_k, & (3.11) \\ & \forall\ l \in \{1,\dots,c\}. \end{aligned}$$

Soient $\mathbf{F}^{(l,i)}$ et $\mathbf{G}^{(l,k)}$ les matrices de taille $(p \times p)$ telles que :

$$\mathbf{F}^{(l,i)} = \sum_{A_j \ni \omega_l} |A_j|^{\alpha-1} m_{ij}^{\beta} \overline{\mathbf{S}}_j \qquad \forall\, l \in \{1,\ldots,c\},\ i \in \{1,\ldots,n\}, \tag{3.12}$$

$$\mathbf{G}^{(l,k)} = \sum_{i=1}^{n} \sum_{A_j \supseteq \{\omega_k,\omega_l\}} |A_j|^{\alpha-2} m_{ij}^{\beta} \overline{\mathbf{S}}_j \qquad \forall\, k,l \in \{1,\ldots,c\}. \tag{3.13}$$

Il est alors possible de former deux nouvelles matrices $\mathbf{F}$ et $\mathbf{G}$ de taille $(cp \times np)$ et $(cp \times cp)$:

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}^{(1,1)} & \mathbf{F}^{(1,2)} & \ldots & \mathbf{F}^{(1,n)} \\ \mathbf{F}^{(2,1)} & \mathbf{F}^{(2,2)} & \ldots & \mathbf{F}^{(2,n)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{F}^{(c,1)} & \mathbf{F}^{(c,2)} & \ldots & \mathbf{F}^{(c,n)} \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} \mathbf{G}^{(1,1)} & \mathbf{G}^{(1,2)} & \ldots & \mathbf{G}^{(1,c)} \\ \mathbf{G}^{(2,1)} & \mathbf{G}^{(2,2)} & \ldots & \mathbf{G}^{(2,c)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}^{(c,1)} & \mathbf{G}^{(c,2)} & \ldots & \mathbf{G}^{(c,c)} \end{pmatrix}. \tag{3.14}$$

En disposant tous les objets $\mathbf{x}_i$ dans un unique vecteur $\mathbf{X}$ de taille $(np \times 1)$ et en réorganisant la matrice $\mathbf{V}$ de façon à former un vecteur de taille $(cp \times 1)$, nous pouvons déterminer $\mathbf{V}$ à partir du système d'équations linéaires suivant :

$$\text{Soient } \mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} \text{ et } \mathbf{V} = \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_c \end{pmatrix}, \quad \text{alors } \mathbf{GV} = \mathbf{FX}. \tag{3.15}$$

Notons qu'au lieu de résoudre p systèmes à c inconnues comme dans le cas d'une métrique Euclidienne, nous devons maintenant résoudre un unique système à cp équations et cp inconnues. Cette complexité plus élevée est le prix à payer pour avoir une métrique qui s'adapte automatiquement à la forme des classes.

Mise à jour des matrices définissant la métrique

Les matrices $\mathbf{M}$ et $\mathbf{V}$ sont maintenant considérées comme fixes afin de pouvoir déterminer pour chaque classe ω_k la matrice $\mathbf{S}_k$ qui minimise la fonction objectif sous les contraintes (3.3). Pour cela, nous introduisons des multiplicateurs de Lagrange. Le Lagrangien associé au problème s'écrit de la manière suivante :

$$\mathcal{L}(\mathbf{S}, \lambda_1, \ldots, \lambda_c) = J_{ECM}(\mathbf{M}, \mathbf{V}, \mathbf{S}) - \sum_{k=1}^{c} \lambda_k \left(\det(\mathbf{S}_k) - \varrho_k\right). \tag{3.16}$$

Rappelons l'équation de la distance entre un objet $\mathbf{x}_i$ et un sous-ensemble A_j :

$$d_{ij}^2 = (\mathbf{x}_i - \overline{\mathbf{v}}_j)^\top \overline{\mathbf{S}}_j (\mathbf{x}_i - \overline{\mathbf{v}}_j) = (\mathbf{x}_i - \overline{\mathbf{v}}_j)^\top \left(\frac{1}{|A_j|} \sum_{k=1}^{c} s_{kj} \mathbf{S}_k \right) (\mathbf{x}_i - \overline{\mathbf{v}}_j). \tag{3.17}$$

Sachant que les dérivées de $\mathbf{x}^\top \mathbf{A} \mathbf{x}$ et $\det(\mathbf{A})$ par rapport à une matrice symétrique $\mathbf{A}$ donnent $\mathbf{x}\mathbf{x}^\top$ et $\det(\mathbf{A})\mathbf{A}^{-1}$, la dérivée du Lagrangien $\mathcal{L}$ par rapport à une matrice $\mathbf{S}_l$ est donnée par l'équation suivante :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{S}_l} = \sum_{i=1}^{n} \sum_{A_j \neq \emptyset} m_{ij}^{\beta} |A_j|^{\alpha-1} s_{lj} (\mathbf{x}_i - \overline{\mathbf{v}}_j)(\mathbf{x}_i - \overline{\mathbf{v}}_j)^\top - \lambda_l \det(\mathbf{S}_l) \mathbf{S}_l^{-1} \qquad \forall\, l \in \{1,\ldots,c\}. \tag{3.18}$$

Soit $\mathbf{\Sigma}_l$ la matrice suivante :

$$\mathbf{\Sigma}_l = \sum_{i=1}^{n} \sum_{A_j \ni \omega_l} m_{ij}^{\beta} |A_j|^{\alpha-1} (\mathbf{x}_i - \overline{\mathbf{v}}_j)(\mathbf{x}_i - \overline{\mathbf{v}}_j)^{\top} \qquad \forall\, l \in \{1, \ldots, c\}. \tag{3.19}$$

Cette matrice peut être considérée comme l'équivalent dans le cadre évidentiel d'une matrice de variance-covariance floue. L'annulation de la dérivée du Lagrangien (3.18) et l'utilisation de (3.19) permet de trouver pour chaque classe ω_l l'équation du multiplicateur de Lagrange λ_l qui minimise la fonction objectif J_{ECM} :

$$\begin{aligned}
\mathbf{\Sigma}_l - \lambda_l \det(\mathbf{S}_l)\mathbf{S}_l^{-1} = 0 \Rightarrow \quad & \mathbf{\Sigma}_l & &= \lambda_l \varrho_l \mathbf{S}_l^{-1}, & (3.20)\\
\Leftrightarrow \quad & \mathbf{\Sigma}_l \mathbf{S}_l & &= \lambda_l \varrho_l \mathbf{I}, & \\
\Rightarrow \quad & \det(\mathbf{\Sigma}_l \mathbf{S}_l) & &= \lambda_l^p \varrho_l^p, & \\
\Leftrightarrow \quad & \det(\mathbf{\Sigma}_l)\varrho_l & &= \lambda_l^p \varrho_l^p, & \\
\Leftrightarrow \quad & \lambda_l & &= \frac{(\varrho_l \det(\mathbf{\Sigma}_l))^{\frac{1}{p}}}{\varrho_l}, & (3.21)
\end{aligned}$$

avec p le nombre d'attributs pour un objet $\mathbf{x}_i$ et $\mathbf{I}$ la matrice identité de dimensions $(p \times p)$. En remplaçant λ_l par son expression (3.21) et en utilisant (3.20), nous obtenons l'équation de mise à jour de la matrice $\mathbf{S}_l$:

$$\mathbf{S}_l = (\varrho_l \det(\mathbf{\Sigma}_l))^{\frac{1}{p}} \mathbf{\Sigma}_l^{-1} \qquad \forall\, l \in \{1, \ldots, c\}. \tag{3.22}$$

Chaque terme $(\mathbf{x}_i - \overline{\mathbf{v}}_j)(\mathbf{x}_i - \overline{\mathbf{v}}_j)^{\top}$ représente une matrice symétrique, semi-définie positive. Il en va alors de même pour leur somme pondérée, et donc pour $\mathbf{\Sigma}_l$, qui est par conséquent bien inversible.

Dans la littérature il faut noter que la plupart du temps, faute de connaissance a priori, les contraintes de volumes ϱ_l sont fixées à 1, pour tout $l \in \{1, \ldots, c\}$. L'équation de $\mathbf{S}_l$ est ainsi simplifiée :

$$\mathbf{S}_l = \det(\mathbf{\Sigma}_l)^{\frac{1}{p}} \mathbf{\Sigma}_l^{-1} \qquad \forall\, l \in \{1, \ldots, c\}. \tag{3.23}$$

La nouvelle procédure de ECM avec une métrique adaptative est résumée par l'algorithme 3.1.

3.1.3 Exemple illustratif

Pour mieux comprendre l'intérêt d'ajouter une métrique adaptative à ECM, un jeu de données ToysDataVert est créé. Ce jeu bidimensionnel est constitué de deux classes, chacune étant un mélange de deux lois normales (cf. tableau 3.1 et figure 3.1).

Algorithme 3.1 : Pseudo-code de l'algorithme ECM avec une métrique adaptative

Entrées : Données $\mathbf{X}$ sous forme vectorielle, c le nombre de classes,
$\varrho_k \ \forall k \in \{1, \ldots, c\}$ les contraintes de volumes (par défaut à 1).
Sorties : Partition crédale $\mathbf{M}$, centres de gravité $\mathbf{V}$, métrique $\mathbf{S}$.
début
  Initialisation aléatoire de c prototypes $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_c)$
  tant que *non convergence* **faire**
    Calculer les nouvelles masses $\mathbf{M}$ avec (3.4) et (3.5),
    Calculer les nouveaux centres de gravité $\mathbf{V}$ en résolvant le système d'équation linéaire défini à l'équation (3.15),
    Calculer les nouvelles matrices $\mathbf{S} = \{\mathbf{S}_1, \ldots, \mathbf{S}_l\}$ en utilisant (3.19) et (3.22).
fin

$\mathcal{N}$		Nb objets	classe
μ	σ		
$(0,0)$	$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$	100	1
$(0,7)$		100	1
$(7,0)$		100	2
$(7,7)$		100	2

Tableau 3.1 – Construction du jeu de données ToysDataVert.

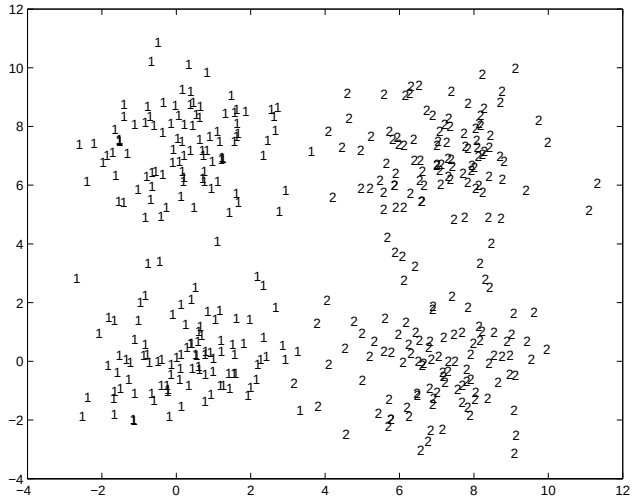


Figure 3.1 – Jeu de données ToysDataVert généré automatiquement à partir de Gaussiennes.

En utilisant la distance Euclidienne avec $\alpha = 1$ et $\delta^2 = 100$, l'algorithme ECM trouve une frontière diagonale entre les classes. La direction de cette diagonale (gauche ou droite) dépend de l'initialisation des centres. Le résultat est présenté à la figure 3.2(a). Les symboles représentent les classes réelles des objets et les différents niveaux de gris les éléments focaux issus de la partition crédale nette pour lesquels les points sont affectés. Les centres des classes sont représentés par des croix de grande taille et les isobarycentres des sous-ensembles non singletons $A_j \subseteq \Omega$ par des étoiles de grande taille. L'affectation des points par la règle du maximum de probabilité pignistique donne un taux d'erreur de près de 25%.

Il n'est donc visiblement pas approprié d'utiliser une distance Euclidienne pour ce type de données. La distance de Mahalanobis semble plus adaptée : l'expérience est donc reproduite en utilisant cette distance avec le même paramétrage que précédemment et $\varrho_k = 1 \ \forall k \in \{1, \ldots, c\}$ (cf. figure 3.2(b)). Selon l'initialisation des centres, la frontière de décision entre les deux classes peut être horizontale ou verticale. La figure montre les ellipses correspondant aux matrices $\mathbf{S}_j$ de chaque sous-ensemble $A_j \subseteq \Omega$.

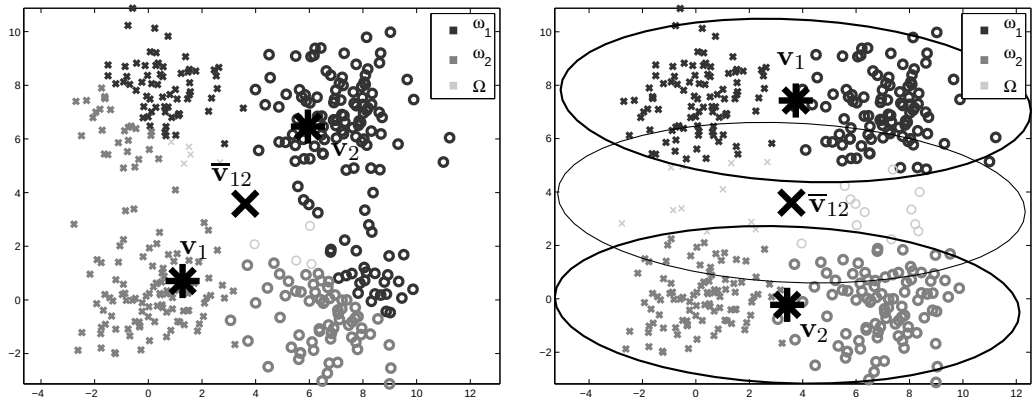


Figure 3.2 – Partitions crédales nettes obtenues avec ECM en utilisant une distance Euclidienne (a) et de Mahalanobis (b) pour le jeu de données ToysDataVert. Les symboles représentent les classes réelles des objets et les différents niveaux de gris correspondent aux sous-ensembles auxquels les objets sont affectés. Les croix et étoiles de grandes tailles représentes les barycentres des sous-ensembles.

Cet exemple montre qu'il existe parfois plusieurs solutions de classification pour un jeu de données. Sans connaissance supplémentaire, il est impossible de décider quel partitionnement est correct. La partie suivante explique comment des contraintes peuvent être intégrées à l'algorithme ECM afin de l'aider à trouver la solution désirée.

3.2 ECM avec intégration de connaissance a priori

La connaissance a priori ajoutée à l'algorithme ECM correspond à des contraintes sur des paires objets. Rappelons qu'une contrainte Must-Link spécifie que deux objets doivent être dans la même classe et une contrainte Cannot-Link indique que deux objets n'appartiennent pas à la même classe. Ci-dessous, nous expliquons comment ces contraintes peuvent être traduites dans le formalisme des fonctions de croyance, puis ajoutées à ECM, afin de créer un nouvel algorithme nommé Constrained-ECM (CECM).

3.2.1 Expression des contraintes

Soient $\mathbf{x}_i$ et $\mathbf{x}_j$ deux objets décrits par deux fonctions de masses m_i et m_j. La fonction de masse conjointe $m_{i\times j}(A \times B)$, exprimée par (2.19) et (2.20), permet de définir l'événement θ "les deux objets appartiennent à la même classe", puis de calculer sa plausibilité conjointe $pl_{i\times j}(\theta)$ (2.23). Dans Ω^2, le complémentaire de θ,

noté $\overline{\theta}$, représente l'événement "$\mathbf{x}_i$ et $\mathbf{x}_j$ n'appartiennent pas à la même classe". La plausibilité correspondante $pl_{i\times j}(\overline{\theta})$ est alors obtenue par :

$$\begin{aligned} pl_{i\times j}(\overline{\theta}) &= 1 - m_{i\times j}(\emptyset) - bel_{i\times j}(\theta), \\ &= 1 - m_{i\times j}(\emptyset) - \sum_{\{A\times B\subseteq\Omega^2 \mid \emptyset\neq(A\times B)\subseteq\theta\}} m_{i\times j}(A\times B) \\ &= 1 - m_{i\times j}(\emptyset) - \sum_{k=1}^{c} m_i(\{\omega_k\})\ m_j(\{\omega_k\}). \end{aligned} \tag{3.24}$$

Prenons par exemple cinq individus à classer dans deux classes. Une partition crédale est donnée par le tableau 3.2. Les objets 1 et 2 ont des degrés de croyances assez similaires, ils sont donc très probablement dans la même classe. Inversement, les objets 1 et 3 ont des masses catégoriques sur deux singletons différents. Ils n'appartiennent donc pas à la même classe. Aucune connaissance n'est disponible sur la classe de l'objet 4 et l'objet 5 est vraisemblablement un objet atypique car il n'appartient à aucune des classes en présence.

A	$m_1(A)$	$m_2(A)$	$m_3(A)$	$m_4(A)$	$m_5(A)$
$\emptyset$	0	0.1	0	0	0.8
$\{\omega_1\}$	1	0.85	0	0	0.1
$\{\omega_2\}$	0	0	1	0	0.1
Ω	0	0.05	0	1	0

Tableau 3.2 – Exemple de partition crédale.

De cette partition crédale, il est possible de déduire les fonctions de masses décrivant l'appartenance conjointe de différents objets, consignées dans le tableau (3.3).

$F = A\times B$	$m_{1\times2}(F)$	$m_{1\times3}(F)$	$m_{1\times4}(F)$	$m_{1\times5}(F)$
$\emptyset\times\emptyset$	0.1	0	0	0.8
$\{\omega_1\}\times\{\omega_1\}$	0.85	0	0	0.1
$\{\omega_1\}\times\{\omega_2\}$	0	1	0	0.1
$\{\omega_1\}\times\Omega$	0.05	0	1	0
$\{\omega_2\}\times\{\omega_1\}$	0	0	0	0
$\{\omega_2\}\times\{\omega_2\}$	0	0	0	0
$\{\omega_2\}\times\Omega$	0	0	0	0
$\Omega\times\{\omega_1\}$	0	0	0	0
$\Omega\times\{\omega_2\}$	0	0	0	0
$\Omega\times\Omega$	0	0	0	0

Tableau 3.3 – Masses quantifiant la croyance sur l'appartenance conjointe entre différents individus.

Les plausibilités conjointes associées aux événements θ et $\overline{\theta}$ sont données par le tableau (3.4). Il est alors possible de remarquer que la valeur de la plausibilité

$pl_{1\times2}(\overline{\theta})$, quasi-nulle, et la valeur de $pl_{1\times2}(\theta)$, proche de 1, permettent de déduire que les individus 1 et 2 sont certainement dans la même classe classe. À l'inverse, $pl_{1\times3}(\theta) = 0$ et $pl_{1\times3}(\overline{\theta}) = 1$ permet d'exprimer la forte possibilité que les objets 1 et 3 soient dans deux classes différentes. Notons qu'il n'est par exemple pas possible de déduire la relation entre les individus $\mathbf{x}_1$ et $\mathbf{x}_4$, car leurs plausibilités associées sont égales à 1. Enfin, si les plausibilités sur les deux événements θ et $\overline{\theta}$ sont faibles pour deux objets particuliers, alors au moins un des deux objets est atypique (c'est le cas ici avec $pl_{1\times5}(F)$).

F	$pl_{1\times2}(F)$	$pl_{1\times3}(F)$	$pl_{1\times4}(F)$	$pl_{1\times5}(F)$
θ	0.9	0	1	0.1
$\overline{\theta}$	0.05	1	1	0.1

Tableau 3.4 – Plausibilités pour les événements θ et $\overline{\theta}$.

3.2.2 Expression de la fonction objectif

Supposons maintenant que la partition crédale est inconnue mais qu'il existe des contraintes sur les individus. Il est alors possible de traduire ces contraintes sous forme de plausibilités $pl_{i\times j}$ et de les utiliser pour trouver une partition crédale. Si $\mathbf{x}_i$ et $\mathbf{x}_j$ sont deux objets contraints par un Must-Link, c'est-à-dire si $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$, alors $pl_{i\times j}(\overline{\theta})$ doit avoir une valeur faible. Inversement, si il existe une contrainte Cannot-Link entre $\mathbf{x}_i$ et $\mathbf{x}_j$ (donc si $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$), alors $pl_{i\times j}(\theta)$ doit avoir une valeur aussi petite que possible. Nous définissons donc le coût J_{CONST} de violer les contraintes Must-Link et Cannot-Link de la manière suivante :

$$J_{CONST}(\mathbf{M}) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} pl_{i\times j}(\overline{\theta}) + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} pl_{i\times j}(\theta). \tag{3.25}$$

Il faut noter que la minimisation globale de ce terme génère deux solutions. La première solution consiste en l'obtention de masses catégoriques sur les mêmes singletons dans le cas d'un Must-Link et deux singletons différents dans le cas d'un Cannot-Link. Une autre solution triviale consiste, pour chaque contrainte, à affecter toute la part de croyance de l'un des deux point contraints dans l'ensemble vide. Ce point est alors considéré comme un objet atypique. Par conséquent, si le second objet contraint n'est pas lui aussi atypique, le coût J_{CONST} reste minimal bien que la contrainte Must-Link ou Cannot-Link ne soit pas respectée. Cette solution non désirée est gérée dans l'algorithme CECM par l'association de J_{ECM} à J_{CONST}. Le terme J_{ECM} comprend en effet l'hyper-paramètre ρ qui permet de contrôler l'importance donnée aux objets atypiques. L'algorithme CECM minimise donc la fonction objectif suivante :

$$J_{CECM}(\mathbf{M}, \mathbf{V}, \mathbf{S}) = (1-\xi)\left(\frac{1}{2^c n} J_{ECM}(\mathbf{M}, \mathbf{V}, \mathbf{S})\right) + \xi\left(\frac{1}{|\mathcal{M}| + |\mathcal{C}|} J_{CONST}(\mathbf{M})\right), \tag{3.26}$$

sous les contraintes (2.17). Les coefficients $\frac{1}{2^c n}$ et $\frac{1}{|\mathcal{M}|+|\mathcal{C}|}$ sont ajoutés afin de normaliser chaque terme. Ainsi, le paramètre $\xi \in [0, 1]$, utilisé pour contrôler l'importance

donnée aux contraintes par rapport au modèle géométrique, peut être fixé à 0.5 si l'expert désire donner le même poids aux deux termes.

3.2.3 Optimisation

L'optimisation de ce nouveau critère consiste, de la même manière que pour l'algorithme ECM avec une métrique adaptative, à minimiser alternativement les matrices **M**, **V** et **S**. Le terme de pénalisation J_{CONST} ne dépendant ni des centres de gravité, ni de la distance entre les objets et les sous-ensembles $A_j \in \Omega$, l'optimisation des matrices **V** et **S** est similaire à J_{ECM} avec une métrique adaptative. Ainsi, les équations de mise à jour de **V** (respectivement **S**) correspondent au système linéaire (3.15) (respectivement aux équations (3.19) et (3.22)).

Le problème est par contre plus complexe pour les fonctions de masses **M**. Une équation directe de mise à jour des m_{ij} à partir des conditions d'optimalité n'est en effet plus possible. Cependant, si l'hyper-paramètre β est fixé à 2, alors la minimisation de la fonction objectif (3.26) par rapport à **M** devient un problème quadratique à contraintes linéaires et la convergence de l'algorithme est assurée en un temps raisonnable. Pour nos expérimentations, nous avons utilisé une implémentation Matlab de la méthode de programmation quadratique développée par Ye et Tse [1][71].

3.2.4 Exemple illustratif

Reprenons l'exemple utilisé dans la partie 3.1.3. Pour le jeu de données Toys-DataVert, l'algorithme ECM utilisant une métrique adaptative trouve deux classes séparées verticalement ou horizontalement, selon l'initialisation des centres de gravité. Dans le pire des cas, la mauvaise solution est sélectionnée, c'est-à-dire ici la solution donnant une frontière horizontale et qui affecte la moitié des objets à la mauvaise classe (cf. figure 3.3(a)). Pour éviter cette solution non désirée, il faut ici avoir connaissance d'informations supplémentaires, sous forme par exemple de contraintes sur des paires d'objets. Dans ce cas, l'ajout de 10 contraintes choisies de manière aléatoire permet de faire converger CECM vers la solution désirée (cf. figure 3.3(b)).

3.3 Protocole expérimental

Dans cette partie, nous évaluons les performances de l'algorithme sur différents jeux de données dont les classes sont déjà connues. Ci-dessous, nous présentons ces jeux de données et leurs caractéristiques, ainsi que la méthode d'évaluation et les différents paramétrages utilisés pour l'algorithme.

1. Le code Matlab est disponible sur le site `http://www.stanford.edu/~yyye/matlab.html`

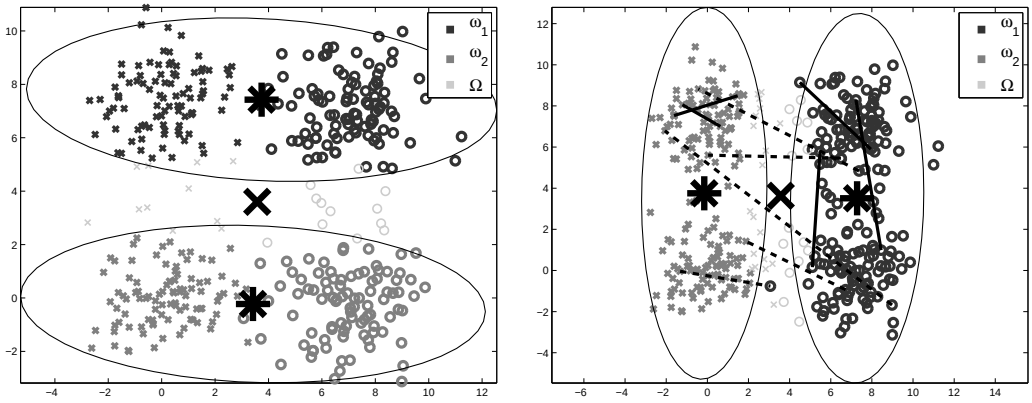


Figure 3.3 – Partitions crédales nettes obtenues pour ToysDataVert : avec ECM en utilisant une distance de Mahalanobis (a), avec CECM en utilisant une distance de Mahalanobis et 10 contraintes. Les lignes continues (respectivement en pointillés) représentent les contraintes Must-Link (respectivement Cannot-Link). Les symboles correspondent aux classes réelles et les niveaux de gris aux sous-ensembles trouvés.

3.3.1 Données

Données de l'UCI

L'Université de Californie à Irvine fournit un ensemble de bases de données à la communauté de fouille de données [2]. Nous avons sélectionné les jeux les plus souvent utilisés dans le domaine de la classification automatique par contraintes. Le tableau (3.5) détaille les caractéristiques de ces différents jeux. Notons que de manière à limiter les temps d'expérimentation, nous avons choisi des jeux où le nombre de classes est limité.

	Iris	Wine	Glass	Ionosphere	LettersIJL
Nombre d'objets n	150	178	214	351	227
Nombre de classes c	3	3	2	2	2
Dimension p	4	13	8	33	16

Tableau 3.5 – Caractéristiques des jeux de données provenant de l'UCI machine learning.

La base de données Iris fait partie des bases les plus employées dans le domaine de la classification par contraintes. Elle est composée de trois classes de 50 spécimens, chacune des classes représentant différentes espèces d'iris : setosa, versicolor et virginica. Il faut noter que les classes ont une distribution non sphérique et qu'une seule des classes est linéairement séparable des deux autres (cf. figure 3.4(a)).

La base de données Wine, utilisée dans [64, 7, 67], est un ensemble de données représentant trois types de vins différents. Les données sont réparties comme suit : la première classe comprend 59 individus, la seconde classe 71 et la dernière 48. Une

2. Disponible à l'URL `http://www.ics.uci.edu/~mlearn`

fois les données normalisées, les classes ont une forme sphérique et sont adjacentes (cf. figure 3.4(b)).

La base de données Glass est un jeu composé de six différents types verres. Nous avons décidé de regrouper ces verres selon leurs origines (cf. figure 3.4(c)). La première classe, qui comprend 163 objets, correspond aux vitres de fenêtres et contient deux types de verres : les vitres pour habitations et celles pour voitures. La seconde classe, constituée de 51 individus, est composée de tous les autres verres qui ne sont pas utilisés comme fenêtres : les verres de containers, la vaisselle et les lampes.

La base de données Ionosphere (cf. figure 3.4(d)) sépare les radars qui détectent un certain type de structure dans l'ionosphère, des radars qui ne le détectent pas. Il s'agit donc d'un problème de classification binaire avec 126 objets dans la première classe et 225 dans la seconde. Ce jeu de données, utilisé dans [7, 67], est réputé pour être difficile. Il est donc intéressant d'ajouter de la connaissance a priori pour améliorer les résultats initiaux de classification.

La base de données LettersIJL est issu du jeu Letters. Ce dernier est composé d'images en noir et blanc. Chaque image représente une lettre en capital parmi les 26 lettres de l'alphabet latin. Ces images sont ensuite converties en données vectorielles en utilisant la méthode des moments sur les différentes caractéristiques des images telles que la largeur et la longueur des lettres. De la même manière que Bilenko & al [7], 10% des données provenant des classes correspondant aux lettres $\{I, J, L\}$ sont sélectionnées. Le nouveau jeu contient alors 81 individus pour la première classe, 72 pour la seconde et 71 pour la dernière (cf. figure 3.4(e)).

Données images

L'intérêt de l'algorithme CECM est aussi illustré sur deux images. La première correspond à une image en niveaux de gris d'un avion, comme le montre la figure 3.5. Le but est de segmenter l'image pour séparer l'avion du fond. Afin de limiter le temps de calcul, un prétraitement consiste à réduire l'image en 161×241 pixels puis à effectuer une quantification de l'image en 200 prototypes à l'aide de la méthode LVQ (Learning Vector Quantization) développée par Kohonen [40]. Un algorithme de classification automatique peut alors être appliqué sur les prototypes, les pixels de l'image étant affectés à la classe du prototype le plus proche.

La seconde image est une image médicale de 156×141 pixels reprise de l'article [8] [3] (cf. figure 3.6). Cette image, qui représente un cerveau atteint d'adrénoleucodystrophie, a été obtenue par résonance magnétique (IRM). Trois régions sont clairement visibles : la première, plus claire que les autres, correspond à la zone pathologique. La zone la plus foncée correspond aux tissus cérébraux normaux et la partie comprenant des niveaux de gris intermédiaires correspond aux ventricules et au liquide cérébro-spinal. Le but étant d'isoler la zone pathologique des autres parties du cerveau, le nombre de classes est fixé à 2. Les niveaux de gris de l'image

3. Nous remercions le professeur Catherine Adamsbaum et le professeur Isabelle Bloch pour nous avoir fourni les images de cerveaux.

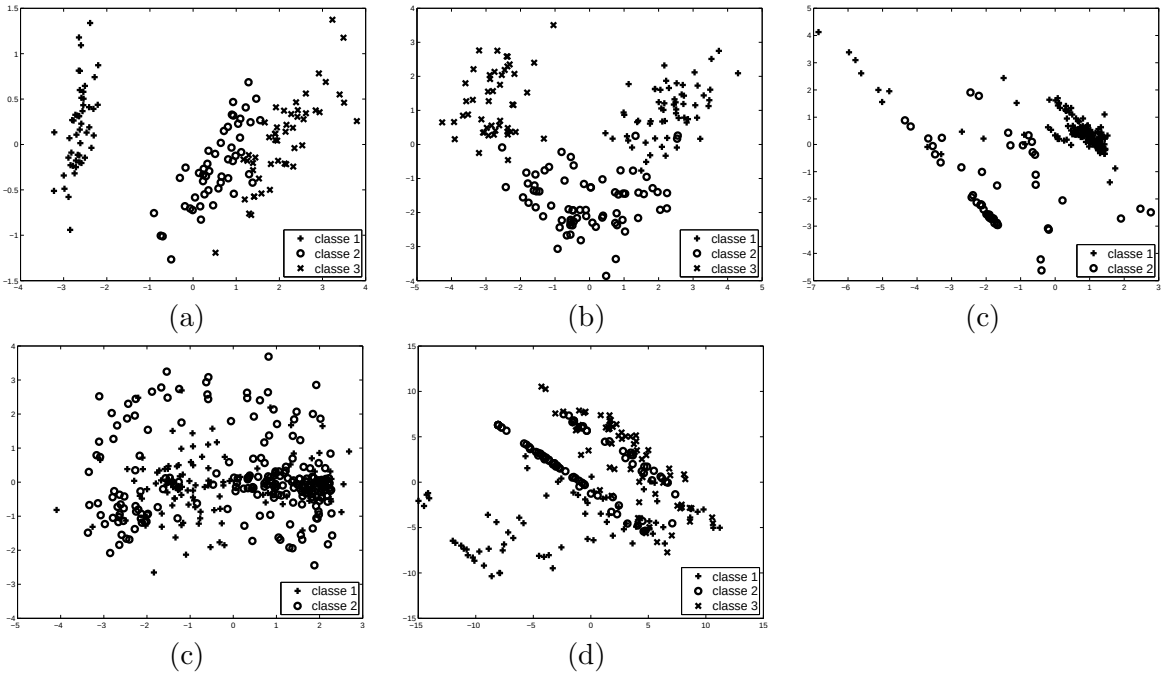


Figure 3.4 – Projection des données sur le premier plan de l'ACP pour les jeux de données Iris (a), Wine (b), Glass (c), Ionosphere (d) et LettersIJL (e). Les pourcentages d'inertie restitués par le premier plan sont 0.98 pour Iris, 0.55 pour Wine, 0.74 pour Glass, 0.44 pour Ionosphere et 0.61 pour LettersIJL.

Figure 3.5 – Image avion.

sont tout d'abord normalisés à 1 puis transformés en 100 objets à l'aide de la même méthode de quantification d'image que précédemment.

3.3.2 Évaluation de la qualité de la classification

Méthode d'évaluation

Afin d'évaluer les performances de CECM sur un jeu de données, il est possible de comparer la partition dure $\widehat{P}$, déterminée en affectant chaque objet à la classe de

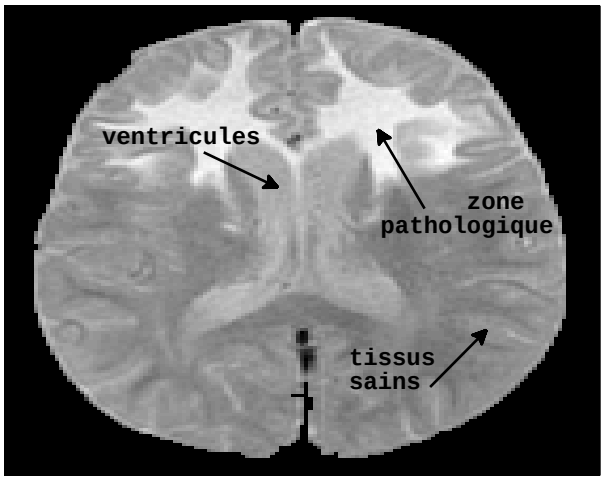


Figure 3.6 – Image de cerveau. La partie la plus claire représente la zone pathologique alors les autres zones correspondent aux tissus sains, aux ventricules et au liquide cérébro-spinal.

probabilité pignistique maximale après convergence de l'algorithme, avec la partition réelle P du jeu initialement connue. Soit a (respectivement b) le nombre de couples d'individus simultanément classés dans la même classe (respectivement dans des classes différentes) par P et $\widehat{P}$. L'indice de Rand (noté RI) permet de mesurer le degré de concordance global de P et $\widehat{P}$:

$$RI = \frac{2(a+b)}{n(n-1)}. \tag{3.27}$$

Il est parfois nécessaire de réaliser plusieurs essais pour une expérience, par exemple si l'initialisation des centres de gravité ou des contraintes est aléatoire. Une moyenne de l'indice de Rand est alors calculée ainsi qu'un intervalle de confiance à 95% (la distribution est supposée normale).

Réglage des hyper-paramètres

Certains hyper-paramètres sont fixés avant l'étude de l'algorithme. Nous cherchons en effet à observer le comportement de l'algorithme CECM lors de l'ajout de contraintes : le nombre de classes est donc fixé a priori bien qu'il pourrait être choisi de la même manière que l'agorithme ECM avec une distance Euclidienne (cf. 2.2.2, partie Choix des hyper-paramètres). De même, nous cherchons à vérifier qu'une contrainte Must-Link (respectivement Cannot-Link) place correctement deux objets dans la même classe (respectivement dans une classe différente). L'utilisation d'une classe dédiée au bruit n'est donc pas utile pour ce type d'expérimentation. Par conséquent, les jeux de données de l'UCI choisis ne comprennent pas d'objets atypiques et l'hyper-paramètre ρ est toujours fixé à une valeur suffisamment élevée. Enfin, l'hyper-paramètre α est fixé à 1 pour éviter d'obtenir une partition qui s'approche d'une partition floue et le volume ϱ_k de chaque classe ω_k est laissé à 1 car nous supposons qu'aucune information n'est disponible à ce sujet.

À l'inverse, l'hyper-paramètre ξ, le nombre de contraintes et plus particulièrement le type de sélection de contraintes représentent des choix importants pour l'étude de l'influence des contraintes sur l'algorithme CECM. Ces paramétrages sont donc détaillés dans les parties suivantes. Notons finalement que pour toutes les futures expériences, les centres de gravité des classes sont initialisés avec les centres de gravité trouvés par l'algorithme FCM.

3.4 Intérêt de l'ajout de contraintes

L'intérêt d'ajouter des contraintes à l'algorithme ECM est étudié de différentes manières. Dans un premier temps, nous analysons le comportement de la méthode selon le nombre de contraintes introduites. Nous proposons ensuite différentes stratégies de sélection de contraintes permettant une amélioration nette de la qualité de la partition finale.

3.4.1 Ajout de contraintes

Données de l'UCI

Dans ces expérimentations, des paires d'objets sont choisies aléatoirement de manière à introduire des ensembles de contraintes Must-Link ou Cannot-Link. L'hyper-paramètre ξ est fixé à 0.5 pour que les deux termes de la fonction objectif aient le même poids. Ainsi, la structure sous-jacente aux données reste aussi importante que le respect des contraintes. Enfin, l'hyper-paramètre ρ^2 est fixé à 1000.

Les figures 3.7, 3.8 et 3.9 montrent l'évolution de l'indice de Rand (moyenné sur 100 essais) par rapport au nombre de contraintes pour les jeux de données Iris, Wine, Glass, Ionosphere et LettersIJL. Toutes ces bases de données utilisent CECM avec une distance de Mahalanobis, à l'exception de Wine qui conserve la distance Euclidienne.

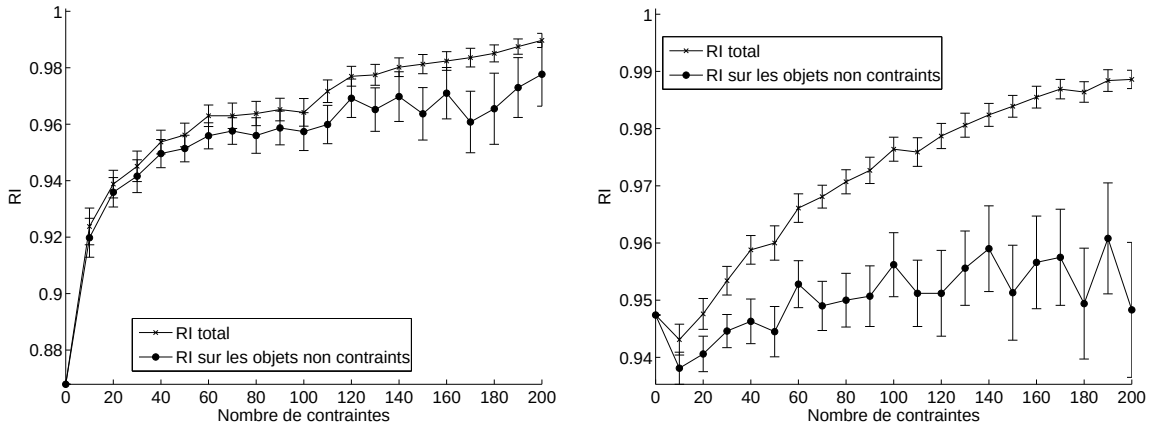


Figure 3.7 – Indices de Rand moyens et intervalles de confiance à 95% trouvés avec l'algorithme CECM tel que $\xi = 0.5$ et $\rho^2 = 1000$, pour Iris (a) et Wine (b).

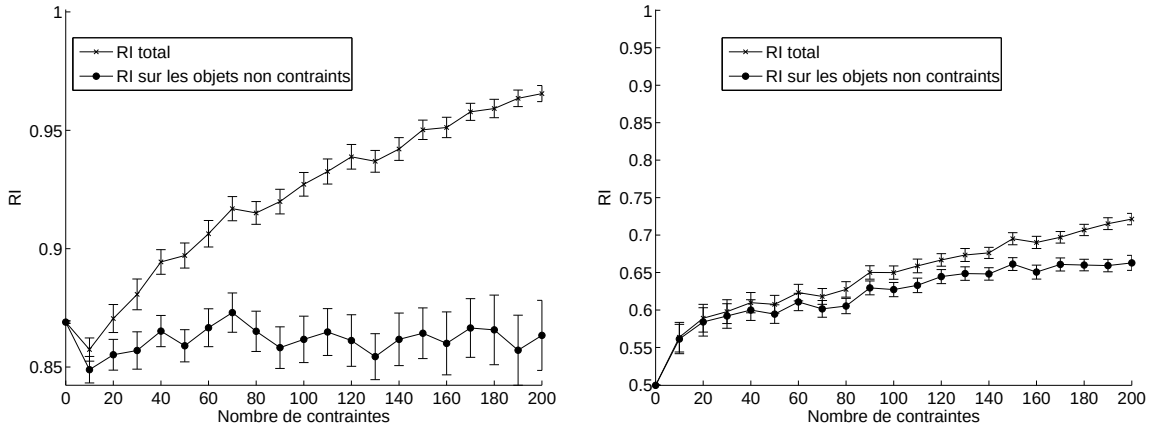


Figure 3.8 – Indices de Rand moyens et intervalles de confiance à 95% trouvés avec l'algorithme CECM tel que $\xi = 0.5$ et $\rho^2 = 1000$, pour Glass (a) et Ionosphere (b).

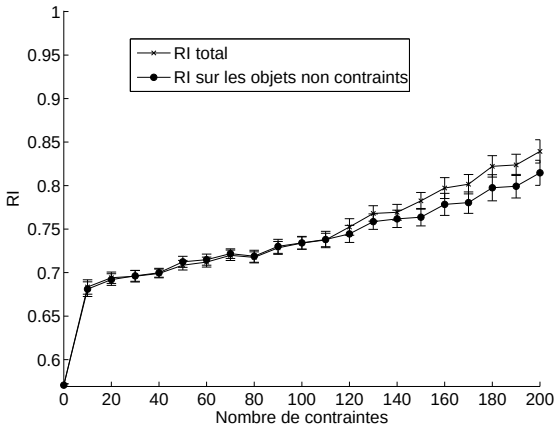
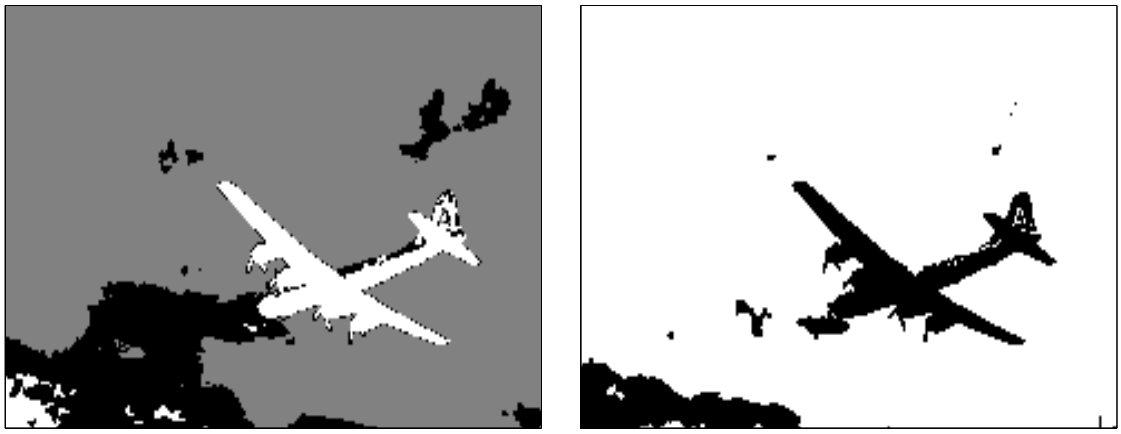


Figure 3.9 – Indices de Rand moyens et intervalles de confiance à 95% trouvés avec l'algorithme CECM tel que $\xi = 0.5$ et $\rho^2 = 1000$, pour LettersIJL.

Nous pouvons observer que l'ajout progressif de contraintes améliore l'indice de Rand global et, pour la plupart des jeux, l'indice de Rand sur les objets non contraints. Ainsi, intégrer des contraintes n'améliore pas seulement la classification des objets contraints : cela permet aussi de guider l'algorithme vers une meilleure solution.

Applications

L'intérêt de CECM est illustré dans le domaine de la segmentation d'images. Une image d'avion (cf. figure 3.5) est transformée en données vectorielles à l'aide d'une méthode de quantification d'image avant d'utiliser l'algorithme ECM avec une distance Euclidienne. Ce dernier, avec $c = 2$, $\alpha = 1$ et $\rho^2 = 10$, trouve la partition crédale dure et la partition pignistique présentées à la figure 3.10. Notons que la distance ρ^2 n'est pas fixée à une valeur élevée car les données ont été normalisées avant l'exécution de ECM. Enfin, les pixels sont affectés à la classe du plus proche prototype.



(a) (b)

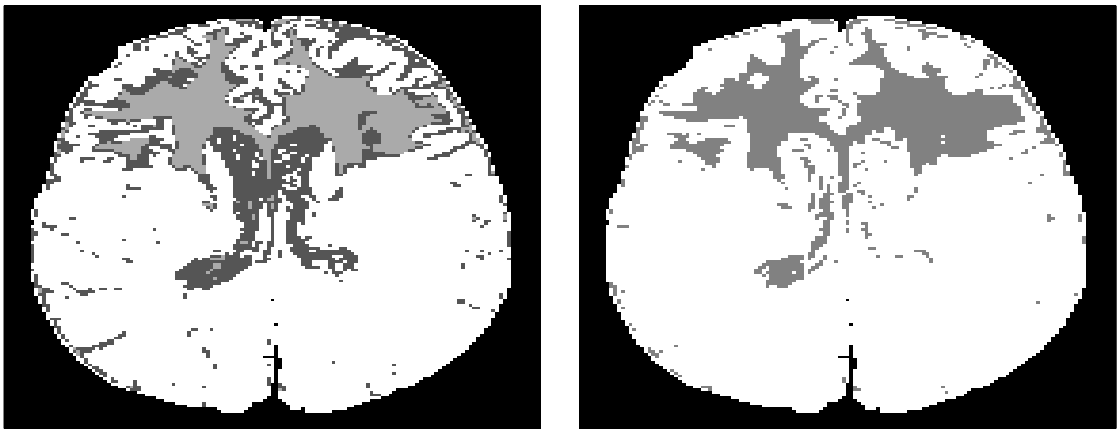
Figure 3.10 – Partition crédale dure (a) et partition calculée à partir des probabilités pignistiques (b) obtenues par ECM avec $\rho^2 = 10$. La partie noir de (a) représente la croyance allouée à l'ensemble Ω.

Il est possible d'observer que l'algorithme ECM n'arrive pas à isoler correctement l'avion du fond. Il existe en effet une vaste région d'incertitude due à la couleur des nuages. Par ailleurs, dans le coin gauche en bas de l'image se trouve une région mal classée. Une seconde expérience est donc effectuée : un utilisateur délimite une région de pixels correspondant à un ensemble de contraintes Must-Link (cf. figure 3.11(a)) et introduit ces nouvelles informations dans l'algorithme CECM utilisant une métrique adaptative. La nouvelle partition crédale est visible à la figure 3.11 (b). Cette dernière montre que les contraintes ont d'une part largement réduit la zone d'incertitude présente dans ECM (les pixels alloués à l'ensemble Ω se trouvent en effet maintenant uniquement à la frontière entre l'avion et le ciel) et d'autre part correctement isolé l'avion du ciel.

(a) (b)

Figure 3.11 – Sélection de contraintes Must-Link pour l'image avion (a) et partition crédale dure obtenues par CECM avec $\rho^2 = 10$ et $\xi = 0.5$ (b). Les pixels noirs de (b) sont des pixels pour lesquels la croyance est allouée à l'ensemble Ω.

La même expérience est maintenant réalisée avec l'image de cerveau décrite à la partie 3.3.1. L'algorithme ECM avec $\alpha = 2$ et $\rho^2 = 10$ permet de trouver la partition crédale dure et les probabilités pignistiques représentées par la figure 3.12. Les régions en blanc et gris clair représentent les deux classes et la zone de couleur gris foncé correspond aux pixels affectés à l'ensemble Ω. Dans un deuxième temps, des



(a) (b)

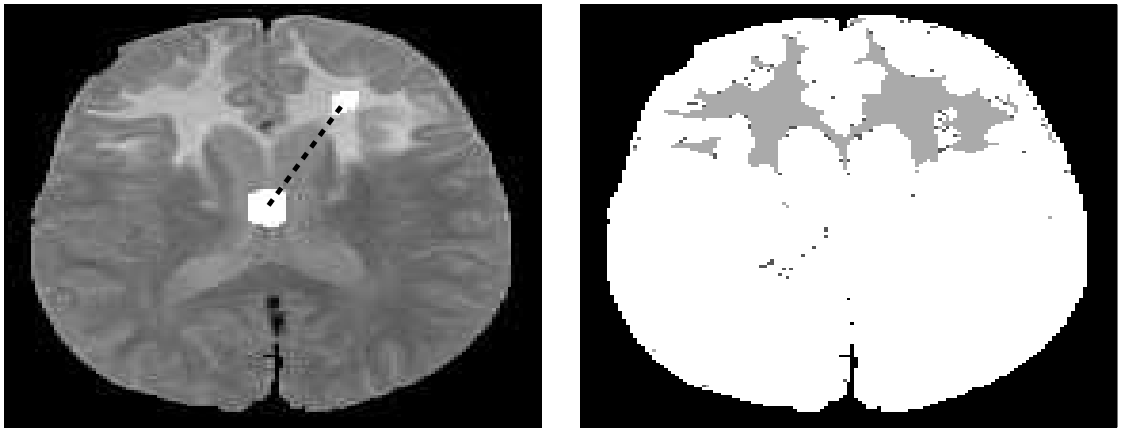
Figure 3.12 – Partition crédale dure (a) et partition calculée à partir des probabilités pignistiques (b) obtenues par ECM avec $\alpha = 2$ et $\rho^2 = 10$. La partie gris foncé de (a) représente la croyance allouée à l'ensemble Ω.

contraintes sont introduites comme le montre la figure 3.13(a). Les parties blanches correspondent aux pixels liés entre eux par une contrainte Must-Link et ces deux parties sont mutuellement liées par une contrainte Cannot-Link. La partition crédale obtenue en appliquant CECM avec une métrique adaptative (et avec $\xi = 0.5$, $\alpha = 2$ et $\rho^2 = 10$) est présentée à la figure 3.13(b). Il est alors possible de constater que les contraintes ont permis de supprimer l'ambiguïté concernant les pixels alloués à l'ensemble Ω et par conséquent ont permis de mieux isoler la zone pathologique du reste du cerveau.

Discussion

Malgré ces résultats probants, il faut noter qu'il existe parfois des ensembles de contraintes qui font décroître les performances de l'algorithme. Cette observation, visible aux figures 3.7(b) et 3.8(a), se vérifie surtout lorsqu'il existe un nombre faible de contraintes. Par ailleurs, le tableau 3.6 reprend les expériences de la partie 3.4.1 (Données de l'UCI) et montre le pourcentage de partitions trouvées par l'algorithme CECM donnant de moins bons résultats que la solution trouvée l'algorithme ECM.

Les contraintes ajoutées dans ces expérimentations étant sans bruit, ces résultats peuvent étonner au premier abord. Dans la partie suivante, nous expliquons les causes de cette chute de performances en prenant des exemples simples. Plusieurs stratégies d'apprentissage actif sont également proposées afin de faire disparaître ce problème.



(a) (b)

Figure 3.13 – Sélection de contraintes Must-Link (zones blanches) et Cannot-Link (lien entre les deux zones blanches) pour l'image cerveau (a) et partition crédale dure obtenue par CECM avec $\alpha = 2$, $\rho^2 = 10$ et $\xi = 0.5$ (b). Les pixels en gris foncé de (b) sont des pixels pour lesquels la croyance est allouée à l'ensemble Ω.

Jeux de	Nb contraintes				
données	10	20	30	40	50
Iris	4%	0%	3%	0%	0%
Wine	65%	46%	26%	26%	13%
Glass	61%	36%	28%	12%	13%
Ionosphere	8%	2%	0%	0%	0%
LettersIJL	0%	0%	0%	0%	0%

Tableau 3.6 – Pourcentage d'échecs de CECM par rapport à ECM.

3.4.2 Apprentissage actif

La technique d'apprentissage actif, décrite au chapitre 1.2.3, sélectionne automatiquement un ensemble pertinent de Q contraintes afin d'obtenir une amélioration de partition la meilleure possible. Pour chaque contrainte, un expert est interrogé sur le lien existant entre les deux objets. Ainsi le nombre de contraintes doit rester faible : il faut en effet limiter le travail de l'expert tout en améliorant de façon notable la qualité de la partition.

Première stratégie d'apprentissage actif : un cas de détérioration de la solution

Dans un premier temps, l'algorithme ECM (avec ou sans métrique adaptative) est exécuté. La partition crédale résultante est ensuite analysée afin de sélectionner une ou plusieurs paires d'objets. Ces paires d'objets doivent être informatives : il est en effet par exemple inutile de choisir deux points affectés à la même classe (grâce à la partition crédale nette) si ils sont très proches l'un de l'autre et inversement, il faut éviter de prendre deux points très éloignés l'un de l'autre si ils se trouvent déjà à l'aide de ECM dans des classes différentes. De plus, les contraintes doivent permettre de lever les doutes planant sur les objets. La première stratégie expérimentée,

qui s'inspire de la méthode de Grira [31], sélectionne alors chaque contrainte de la manière suivante :

- le premier point choisi est un point proche de la frontière. En d'autres termes, cet objet a un fort degré d'incertitude, c'est-à-dire que la majorité de la croyance sur son appartenance est allouée à un ou plusieurs sous-ensembles $A_j \in \Omega$ non singleton. Le point choisi est donc celui qui a la mesure de non-spécificité la plus élevée.
- Le second point est le point le plus proche du premier point qui n'est pas dans la même classe. Pour savoir si deux objets sont ou ne sont pas dans la même classe, nous utilisons la partition dure trouvée à partir du maximum de probabilité pignistique.

Les expériences sont réalisées sur les jeux de données de l'UCI. Les étiquettes des individus étant déjà connues, la procédure de questionnement d'un expert est automatisée. Au départ, un ensemble de 10 contraintes est sélectionné à partir de la partition crédale de ECM. L'algorithme CECM est ensuite exécuté et la partition crédale trouvée permet d'acquérir 10 nouvelles contraintes qui sont ajoutées à l'ensemble précédent de contraintes. Ce procédé est ré-itéré jusqu'à l'obtention d'un ensemble de 200 contraintes.

Il faut noter qu'obtenir 200 contraintes par apprentissage actif n'est pas envisageable dans le cas d'une application réelle. En effet, l'expert doit interagir le moins possible avec l'application, principalement pour deux raisons : cela est coûteux en temps pour l'utilisateur et plus il existe de questions, plus l'attention de expert diminue et par conséquent plus il risque de faire des erreurs. L'idée principale de nos expériences consiste donc uniquement à comparer l'apprentissage actif avec l'ajout de contraintes aléatoires.

La méthode d'apprentissage actif expérimentée donne pour la majorité des jeux de données de l'UCI de moins bonnes performances que la sélection aléatoire de contraintes. La figure 3.14 présente les résultats pour le jeu de données Wine. Tout d'abord, il est possible de remarquer que la courbe pour les objets non contraints ne dépasse pas 160 contraintes : à partir de cette valeur, il n'existe en effet plus de points non contraints. De plus, comme ECM est initialisé au départ avec les centres de gravité trouvés par FCM et que l'apprentissage actif n'entraîne aucune sélection aléatoire de contraintes, la figure 3.14 ne présente pas d'intervalle de confiance. Enfin, nous pouvons observer que l'indice de Rand sur les objets non contraints augmente alors que l'indice de Rand total diminue. Les objets contraints font donc chuter les performances.

Afin de mieux comprendre pourquoi l'ajout de contraintes peut amener ce type de résultat, une nouvelle expérience est réalisée. La figure 3.15(a) montre le premier plan d'ACP de Wine et la partition crédale obtenue par l'algorithme ECM avec $\rho^2 = 1000$ pour ce jeu de données. Une contrainte est alors choisie (cf. figure 3.15(b)) de la même manière que pour les expériences précédentes : le premier objet $\mathbf{o}_1$ est un point très incertain (donc proche des centres de gravité des sous-ensembles de fortes cardinalité) et le second objet $\mathbf{o}_2$ est le point le plus proche de $\mathbf{o}_1$ qui se trouve

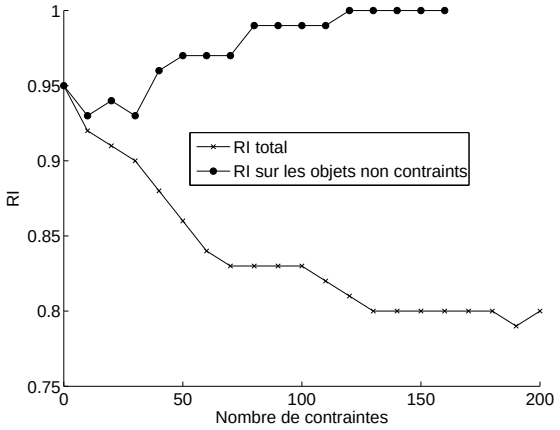


Figure 3.14 – Apprentissage actif inspiré par la méthode de Grira pour Wine.

dans une classe différente (le nombre d'attributs étant élevé pour le jeu de données Wine, l'échelle des distances des figures 3.15(a) et (b) ne reflètent pas exactement la réalité). Il faut noter que les deux objets, nouvellement contraints par un Must-Link, ne sont pas affectés par la partition crédale nette de ECM à la même classe : l'objet $\mathbf{o}_2$ est en effet mal classé.

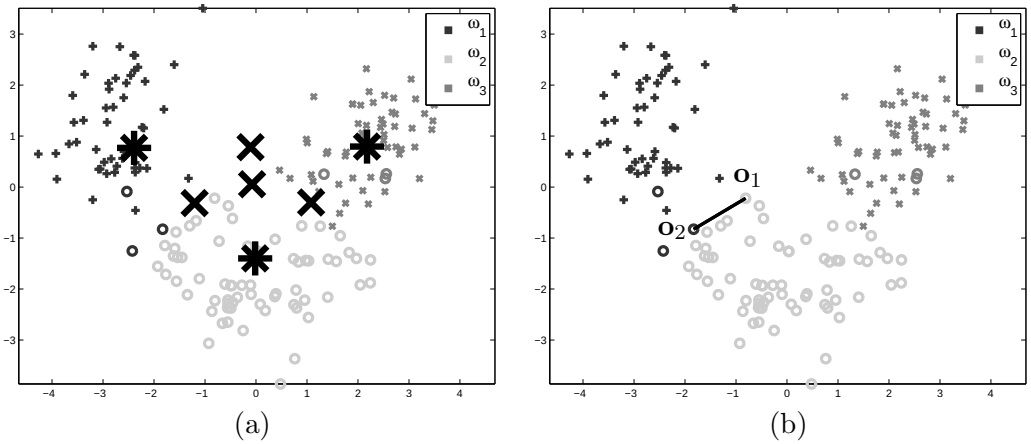


Figure 3.15 – Projection des données Wine sur le premier plan d'ACP. Les figures montrent la partition crédale obtenue par ECM avec $\rho^2 = 1000$. La figure (a) représente les centres de gravité par les croix de grandes tailles pour les singletons et les étoiles de grandes tailles pour les autres sous-ensembles. La figure (b) présente une contrainte sélectionnée. Les symboles correspondent aux classes réelles des objets et les niveaux de gris l'affectation des objets aux différents sous-ensemble de Ω.

L'algorithme CECM est ensuite exécuté et la partition crédale nette résultante est montrée à la figure 3.16. Contrairement à ce qui est souhaité, l'objet $\mathbf{o}_2$ est resté mal classé et c'est l'objet $\mathbf{o}_1$ qui a changé de classe. En étudiant les fonctions de masses attribuées aux objets $\mathbf{o}_1$ et $\mathbf{o}_2$ après ECM, nous pouvons voir que $\mathbf{o}_2$, bien que caractérisé par une forte non-spécificité, a une masse de croyance plus élevée au niveau des singletons que $\mathbf{o}_1$ (cf. tableau 3.4.2). C'est donc cet objet qui va guider l'autre objet vers une nouvelle solution. Le paramètre ξ ayant une valeur très

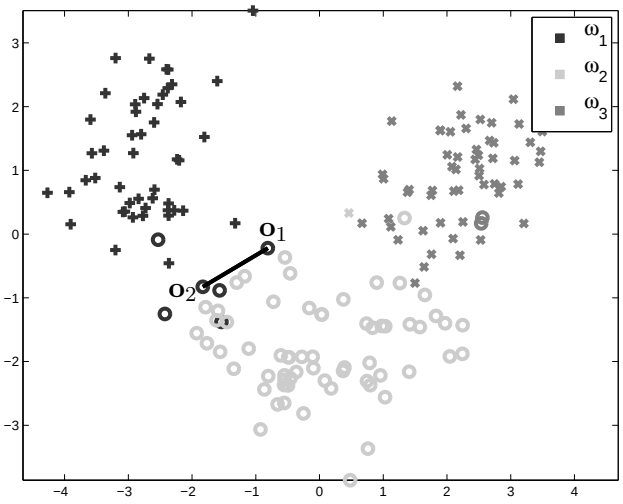


Figure 3.16 – Partition crédale obtenue par CECM avec $\rho^2 = 1000$, $\xi = 0.5$ et une contrainte Must-Link pour Wine. Les symboles correspondent aux classes réelles des objets et les niveaux de gris l'affectation des objets aux différents sous-ensemble de Ω.

élevée pour ce jeu de données (c'est-à-dire $\xi = 0.5$), les masses des points contraints tendent vers une fonction de masse catégorique. Comme l'objet $\mathbf{o}_2$ a initialement plus de croyance allouée à la classe ω_1 qu'à la classe ω_2, l'algorithme CECM accentue ce déséquilibre, qui se répercute sur la fonction de masse associée à l'objet $\mathbf{o}_1$.

	ECM		CECM	
A	$m_1(A)$	$m_2(A)$	$m_1(A)$	$m_2(A)$
$\emptyset$	0	0	0	0
ω_1	0.2	0.26	1	1
ω_2	0.21	0.22	0	0
$\{\omega_1, \omega_2\}$	0.12	0.16	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Ω	0	0	0	0

Tableau 3.7 – Fonctions de masses obtenues avec ECM et CECM pour deux points de Wine (contraints par un Must-Link dans le cas de CECM).

De plus, notons que la masse des points contraints ayant été fortement modifiée, les centres de gravité sont déplacés et il en résulte une modification des masses des points non contraints. Ainsi pour la figure 3.16, le centre de gravité de la classe ω_1 a été attiré par les deux objets contraints, ce qui a permis d'ajouter deux objets non contraints à la classe, alors qu'ils étaient précédemment correctement classés. Notons que ce déplacement des centres de gravité n'est pas visible à l'oeil nu, c'est pourquoi les isobarycentres ne sont pas représentés sur la figure 3.16.

Seconde stratégie d'apprentissage actif

Ces observations ont permis de développer une nouvelle stratégie d'apprentissage actif décrite par l'algorithme 3.2. Le premier point sélectionné reste toujours un

point classé avec un fort degré d'incertitude, cependant le second point correspond maintenant à un objet classé de manière très certaine. Il existe plusieurs techniques pour trouver de tels objets à l'aide de la partition crédale ou des centres de gravité. Nous proposons pour le premier point de repérer, comme avant, l'objet ayant un fort degré de non-spécificité et pour le deuxième point de choisir celui qui se trouve être le plus près d'un des centres. Cette stratégie de sélection a l'avantage de faire disparaître les problèmes d'objets contraints mal classés.

Algorithme 3.2 : Pseudo-code de l'apprentissage actif pour CECM

Entrées : Partition crédale $\mathbf{M}$, centres de gravité $\mathbf{V}$, données $\mathbf{X}$ sous forme vectorielle et Q le nombre de contraintes
Sorties : E l'ensemble des contraintes
début
  $E \leftarrow \{\}$
  tant que *le nombre de contraintes n'est pas égal à* Q **faire**
    Choix du premier point $\mathbf{x}_i$ tel que $\mathbf{x}_i \notin E$ et

$$\mathbf{x}_i = \{i = \arg\max_{i'} N(m_{i'})\}.$$

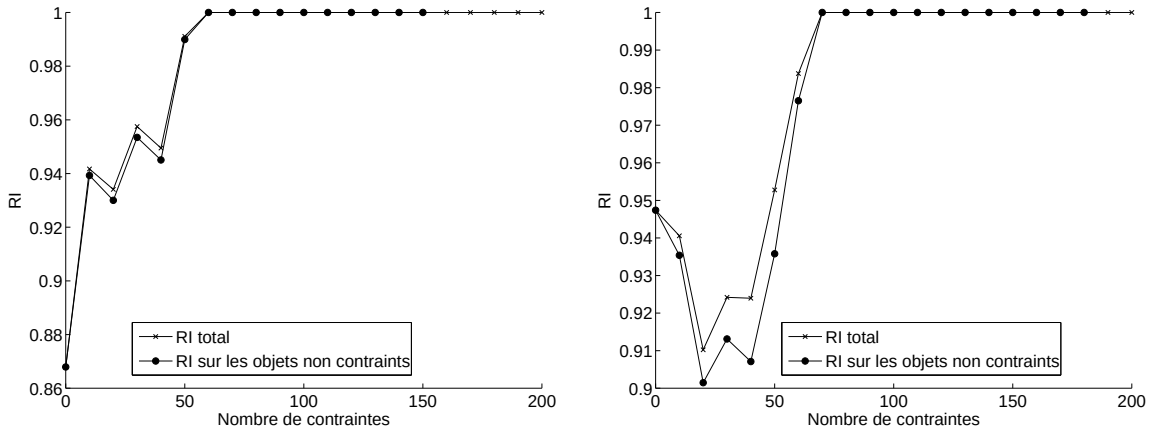
    Choix du second point $\mathbf{x}_j$:

$$\mathbf{x}_j = \{j = \arg\min_{l_k} d(\mathbf{x}_i, \mathbf{x}_{l_k})\} \text{ avec } l_k = \arg\min_{l'} d(\mathbf{x}_{l'}, \mathbf{v}_k)\ \forall k \in \{1 \dots c\}.$$

    Ajout de la paire $(\mathbf{x}_i, \mathbf{x}_j)$ dans l'ensemble E
fin

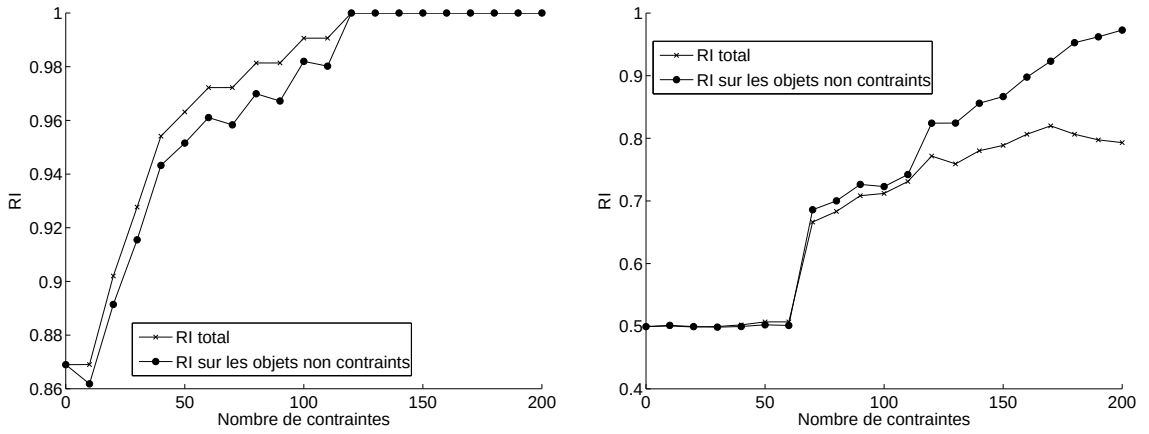
Le protocole expérimental utilisé pour cette méthode d'apprentissage actif est le même que précédemment. Les résultats sont présentés aux figures 3.17, 3.18 et 3.19. Afin de mieux observer l'amélioration des résultats par rapport à une sélection aléatoire de contraintes, l'annexe D présente dans un même graphe l'apprentissage actif et l'ajout de contraintes aléatoire. Nous remarquons à l'aide de ces diverses figures que pour la plupart des jeux de données, la partition réelle finit par être trouvée à partir d'un certain nombre de contraintes. Les résultats sont donc en général bien meilleurs avec l'utilisation de l'apprentissage actif. Il peut cependant toujours exister une chute de performances, notamment pour un nombre peu important de contraintes. C'est le cas par exemple pour le jeu de données Wine (cf. figure 3.17(b)).

Cette chute, bien qu'importante, est corrigée par l'ajout de nouvelles contraintes. La dégradation des performances est en effet due uniquement aux objets non contraints et ceux-ci étant affectés aux classes de manière moins certaine que les objets contraints, leurs fonctions de masses sont plus facilement modifiables. Prenons par exemple l'ajout de 20 contraintes à la base de données Wine. La diminution de la qualité de la partition par rapport à ECM est très élevée (cf. figure 3.17(b)). Pour en comprendre la cause, les partitions crédales nettes pour 0 et 20 contraintes sont présentées à la figure 3.20. Les graphiques permettent en effet de visualiser la tendance qu'ont les centres de gravité à se déplacer vers les isobarycentres des points



(a) (b)

Figure 3.17 – Indices de Rand trouvés par l'algorithme CECM avec $\xi = 0.5$, $\rho^2 = 1000$ et apprentissage actif, pour Iris (a) et Wine (b).



(a) (b)

Figure 3.18 – Indices de Rand trouvés par l'algorithme CECM avec $\xi = 0.5$, $\rho^2 = 1000$ et apprentissage actif, pour Glass (a) et Ionosphere (b).

contraints. L'hyper-paramètre ξ étant fixé à une valeur élevée, les points contraints sont affectés à une classe de manière très certaine. Le mouvement vers la gauche des centres de gravité des classes ω_1 et ω_3 implique une modification nette des fonctions de croyance des objets situés à la frontière des deux classes : auparavant bien classés, ils appartiennent maintenant de manière un peu plus certaine à la classe ω_3. Cette situation sera de nouveau modifiée si de nouvelles contraintes sont ajoutées dans cette zone. Cela permettra en effet de redéplacer les deux centres de gravité vers la droite. C'est en effet ce qui se passe avec l'apprentissage actif pour 50, 60 et 70 contraintes.

Pour conclure, la chute de performances de CECM peut être due à deux types de comportement de l'algorithme :

- un ou plusieurs points contraints sont mal classés. Ce phénomène, irréversible pour une valeur de ξ trop forte, est dû à l'usage d'un couple de points peu certains. Il entraîne ensuite la mauvaise classification d'autres points, qu'ils

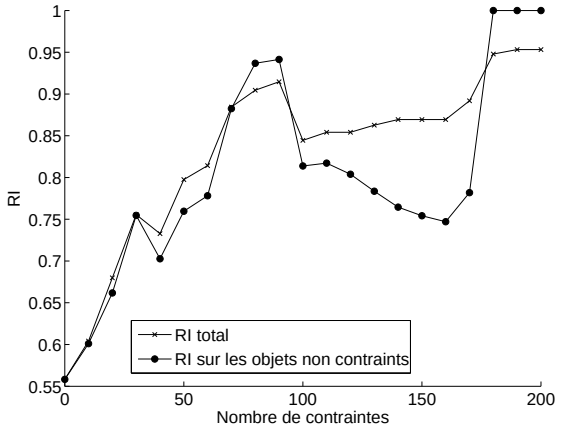
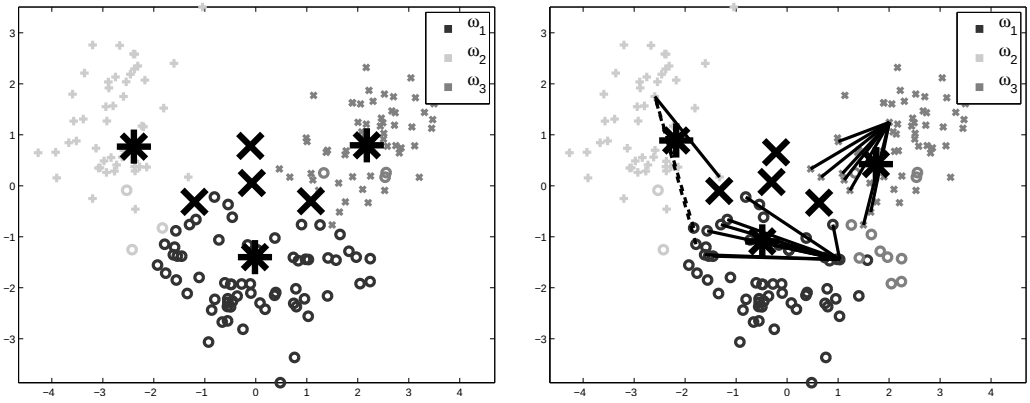


Figure 3.19 – Indice de Rand trouvé par l'algorithme CECM avec $\xi = 0.5$, $\rho^2 = 1000$ et apprentissage actif, pour LettersIJL.



(a) (b)

Figure 3.20 – Partitions crédales nettes de la base Wine obtenues par ECM tel que $\rho^2 = 1000$ (a) et par CECM avec 20 contraintes tel que $\xi = 0.5$, $\rho^2 = 1000$ (b). Les contraintes Must-Link (respectivement Cannot-Link) sont représentées par des lignes continues (respectivement en pointillés). Croix et étoiles de grande taille correspondent aux centres de gravité des classes et des sous-ensembles de Ω. Les symboles représentent les classes réelles des objets.

soient eux-mêmes contraints ou non.

- un ou plusieurs points non contraints auparavant bien classés sont maintenant mal classés. Ceci est la conséquence d'un déplacement trop brusque d'un centre de gravité dû à une importante quantité de points contraints situés dans une région spécifique de l'espace associé à la base de données. Ce phénomène correspond à un problème d'exploitation des données sans exploration (cf. paragraphe discussion de la partie 1.2.3).

3.5 Évaluation de la méthode

Le comportement général de l'algorithme ayant été étudié, nous nous intéressons maintenant au choix de la valeur de ξ.

3.5.1 Choix des hyper-paramètres

L'hyper-paramètre ξ contrôle le compromis entre les contraintes par paires et l'ajustement du modèle géométrique aux données. Plusieurs expérimentations ont été effectuées avec différents paramétrages de ξ et du nombre de contraintes. Ces expérimentations sont effectuées sur les jeux de données de l'UCI et les résultats sont présentés dans les tables 3.8 et 3.9. Comme la manière de sélectionner les contraintes est aléatoire, les résultats sont moyennés sur 100 expériences.

C	ξ	Iris (Mah.)	Wine (Eucl.)	Glass (Mah.)	Ionosphere (Mah.)	LettersIJL (Mah.)
20	0	0,87 ± 0,00	0,95 ± 0,00	0,85 ± 0,00	0,50 ± 0,00	0,57 ± 0,00
	0,1	0,93 ± 0,00	0,95 ± 0,00	0,87 ± 0,01	0,58 ± 0,05	0,71 ± 0,04
	0,2	0,94 ± 0,01	0,95 ± 0,00	0,86 ± 0,01	0,58 ± 0,05	0,70 ± 0,02
	0,3	0,94 ± 0,01	0,95 ± 0,00	0,86 ± 0,01	0,59 ± 0,06	0,69 ± 0,02
	0,4	0,94 ± 0,01	0,95 ± 0,00	0,87 ± 0,01	0,60 ± 0,06	0,69 ± 0,02
	0,5	0,94 ± 0,01	0,95 ± 0,00	0,87 ± 0,01	0,61 ± 0,06	0,69 ± 0,02
	0,6	0,94 ± 0,00	0,95 ± 0,00	0,87 ± 0,01	0,59 ± 0,06	0,70 ± 0,02
	0,7	0,94 ± 0,01	0,95 ± 0,00	0,86 ± 0,01	0,61 ± 0,06	0,69 ± 0,02
	0,8	0,94 ± 0,01	0,95 ± 0,00	0,86 ± 0,01	0,59 ± 0,06	0,69 ± 0,02
	0,9	0,94 ± 0,01	0,95 ± 0,00	0,87 ± 0,01	0,60 ± 0,06	0,69 ± 0,02
50	0	0,87 ± 0,00	0,95 ± 0,00	0,85 ± 0,00	0,50 ± 0,00	0,57 ± 0,00
	0,1	0,96 ± 0,00	0,95 ± 0,00	0,89 ± 0,00	0,61 ± 0,05	0,76 ± 0,03
	0,2	0,96 ± 0,00	0,96 ± 0,00	0,89 ± 0,00	0,62 ± 0,04	0,75 ± 0,04
	0,3	0,96 ± 0,00	0,96 ± 0,00	0,89 ± 0,01	0,62 ± 0,04	0,72 ± 0,02
	0,4	0,96 ± 0,00	0,96 ± 0,00	0,90 ± 0,01	0,61 ± 0,04	0,72 ± 0,02
	0,5	0,96 ± 0,00	0,96 ± 0,00	0,90 ± 0,01	0,62 ± 0,04	0,70 ± 0,02
	0,6	0,96 ± 0,00	0,96 ± 0,00	0,90 ± 0,00	0,61 ± 0,04	0,70 ± 0,02
	0,7	0,96 ± 0,00	0,96 ± 0,00	0,89 ± 0,01	0,61 ± 0,04	0,70 ± 0,02
	0,8	0,96 ± 0,00	0,96 ± 0,00	0,89 ± 0,01	0,61 ± 0,04	0,70 ± 0,02
	0,9	0,96 ± 0,00	0,96 ± 0,00	0,90 ± 0,01	0,61 ± 0,04	0,70 ± 0,02

Tableau 3.8 – Indice de Rand moyen et intervalle de confiance en fonction de ξ pour 20 et 50 contraintes choisies aléatoirement pour les jeux de données de l'UCI.

Nous constatons que le choix de ξ, bien qu'il ait un impact sur l'indice de Rand, n'est pas critique. Les résultats sont effet très stables surtout pour Iris, Wine et Glass et dépendent essentiellement du nombre de contraintes introduites. Ainsi, choisir $\xi = 0.5$ mène généralement à des résultats acceptables dans la plupart des cas.

3.5.2 Robustesse aux contraintes bruitées

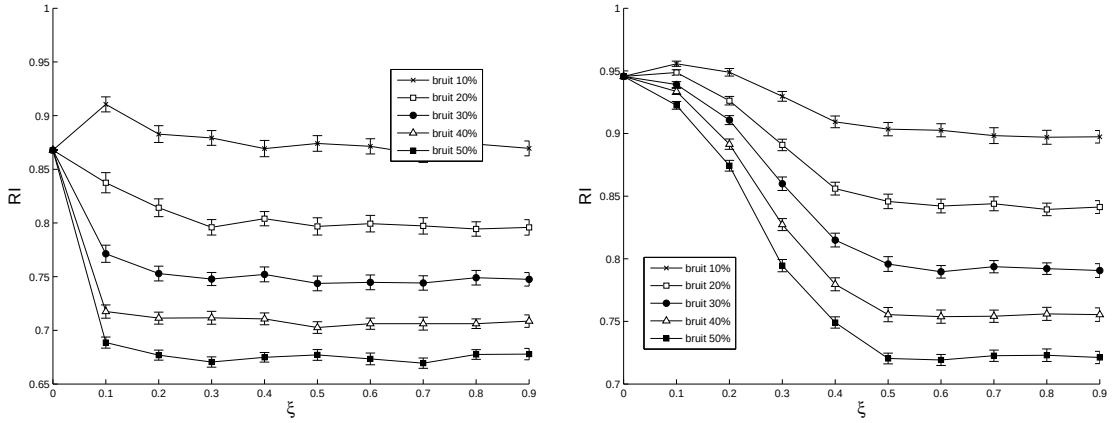
Les contraintes passées à l'algorithme peuvent être bruitées, c'est-à-dire être incohérentes les unes par rapport aux autres. Cette situation d'incohérence se produit lorsqu'au moins une contrainte a été mal définie (par exemple quand une contrainte

C	ξ	Iris (Mah.)	Wine (Eucl.)	Glass (Mah.)	Ionosphere (Mah.)	LettersIJL (Mah.)
100	0	0,87 ± 0,00	0,95 ± 0,00	0,85 ± 0,00	0,50 ± 0,00	0,57 ± 0,00
	0,1	0,97 ± 0,00	0,96 ± 0,00	0,92 ± 0,00	0,65 ± 0,02	0,76 ± 0,01
	0,2	0,97 ± 0,00	0,97 ± 0,00	0,92 ± 0,00	0,64 ± 0,01	0,83 ± 0,01
	0,3	0,97 ± 0,00	0,97 ± 0,00	0,92 ± 0,00	0,65 ± 0,01	0,79 ± 0,01
	0,4	0,97 ± 0,00	0,97 ± 0,00	0,93 ± 0,00	0,65 ± 0,01	0,77 ± 0,01
	0,5	0,97 ± 0,00	0,98 ± 0,00	0,93 ± 0,00	0,65 ± 0,01	0,74 ± 0,01
	0,6	0,97 ± 0,00	0,98 ± 0,00	0,94 ± 0,00	0,65 ± 0,01	0,72 ± 0,01
	0,7	0,97 ± 0,00	0,98 ± 0,00	0,93 ± 0,00	0,65 ± 0,01	0,71 ± 0,01
	0,8	0,97 ± 0,00	0,98 ± 0,00	0,93 ± 0,00	0,65 ± 0,01	0,71 ± 0,00
	0,9	0,97 ± 0,00	0,98 ± 0,00	0,93 ± 0,00	0,64 ± 0,01	0,70 ± 0,01
200	0	0,87 ± 0,00	0,95 ± 0,00	0,85 ± 0,00	0,50 ± 0,00	0,57 ± 0,00
	0,1	0,99 ± 0,00	0,97 ± 0,00	0,94 ± 0,00	0,79 ± 0,03	0,76 ± 0,01
	0,2	0,99 ± 0,00	0,98 ± 0,00	0,96 ± 0,00	0,74 ± 0,01	0,86 ± 0,00
	0,3	0,99 ± 0,00	0,98 ± 0,00	0,96 ± 0,00	0,72 ± 0,01	0,90 ± 0,01
	0,4	0,99 ± 0,00	0,99 ± 0,00	0,96 ± 0,00	0,72 ± 0,01	0,89 ± 0,01
	0,5	0,99 ± 0,00	0,99 ± 0,00	0,97 ± 0,00	0,72 ± 0,01	0,85 ± 0,01
	0,6	0,99 ± 0,00	0,99 ± 0,00	0,97 ± 0,00	0,72 ± 0,01	0,79 ± 0,01
	0,7	0,99 ± 0,00	0,99 ± 0,00	0,97 ± 0,00	0,73 ± 0,01	0,75 ± 0,01
	0,8	0,99 ± 0,00	0,99 ± 0,00	0,97 ± 0,00	0,73 ± 0,01	0,74 ± 0,01
	0,9	0,98 ± 0,00	0,99 ± 0,00	0,97 ± 0,00	0,72 ± 0,01	0,72 ± 0,01

Tableau 3.9 – Indice de Rand moyen et intervalle de confiance en fonction de ξ pour 100 et 200 contraintes choisies aléatoirement pour les jeux de données de l'UCI.

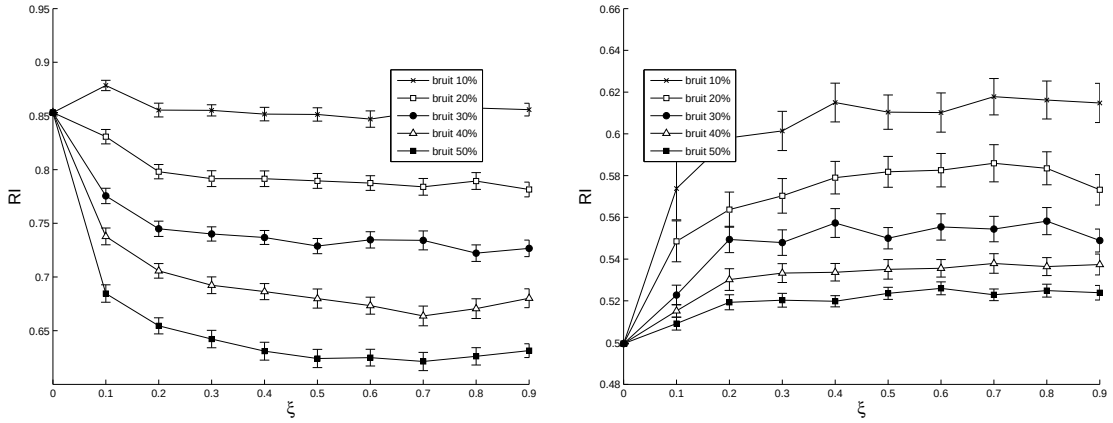
Must-Link a été définie comme une contrainte Cannot-Link). Il est donc important de connaître le comportement de CECM face à cette situation. Pour cela, le nombre de contraintes est fixé à 100 et du bruit est ajouté aléatoirement. Ainsi, une contrainte Must-Link (respectivement Cannot-Link) est changée en contrainte Cannot-Link (respectivement Must-Link) avec une probabilité comprise entre 0.1 et 0.5. De la même manière qu'auparavant, 100 expériences sont lancées pour chaque valeur de ξ et chaque pourcentage de bruit. Les figures 3.21, 3.22 et 3.23 présentent l'indice de Rand en fonction de l'hyper-paramètre ξ.

Nous pouvons remarquer que la robustesse de la méthode dépend essentiellement du jeu de données utilisé : pour Ionosphere et LettersIJL nous obtenons une amélioration des résultats malgré les contraintes bruités alors que les autres jeux de données (Iris, Wine et Glass) sont plus sensibles au bruit. Il est donc essentiel de détecter les contradictions entre les contraintes afin soit de les supprimer, soit d'utiliser une valeur faible de ξ. Une première stratégie consiste à examiner l'évolution de la valeur de chaque terme de la fonction objectif pour ξ variant de 0.1 à 0.9. En effet, si $J_{CECM} > J_{ECM}$ quel que soit ξ, alors la valeur du terme J_{CONST} est élevée, ce qui signifie que les contraintes ont du mal à être toutes respectées. Dans ce cas, il existe sûrement des contradictions au sein même des ensembles de contraintes. Ces contradictions étant significatives, il vaut mieux choisir de ne pas prendre en compte



(a) (b)

Figure 3.21 – Indice de Rand moyen en fonction de ξ pour 100 contraintes choisies aléatoirement et bruitées entre 10% et 50%, pour les jeux de données Iris (a) et Wine (b).



(a) (b)

Figure 3.22 – Indice de Rand moyen en fonction de ξ pour 100 contraintes choisies aléatoirement et bruitées entre 10% et 50%, pour les jeux de données Glass (a) et Ionosphere (b).

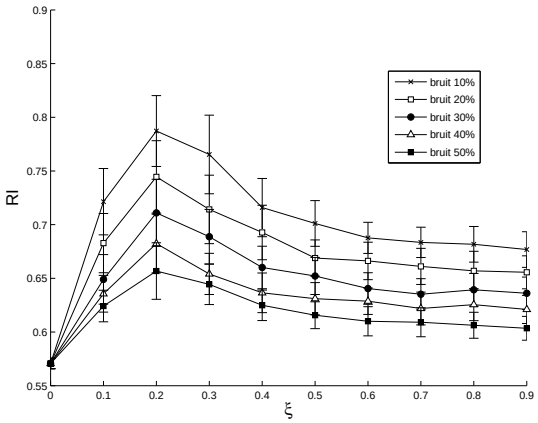


Figure 3.23 – Indice de Rand moyen en fonction de ξ pour 100 contraintes choisies aléatoirement et bruitées entre 10% et 50%, pour le jeu de données LettersIJL.

les contraintes (et donc fixer ξ à 0).

Une seconde solution consiste à effectuer une analyse préliminaire des données afin de vérifier la cohérence des contraintes. Ainsi par exemple, une relation ternaire telle que $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$, $(\mathbf{x}_j, \mathbf{x}_l) \in \mathcal{M}$ et $(\mathbf{x}_l, \mathbf{x}_i) \in \mathcal{C}$ est aberrante. Ces contraintes doivent donc être supprimées.

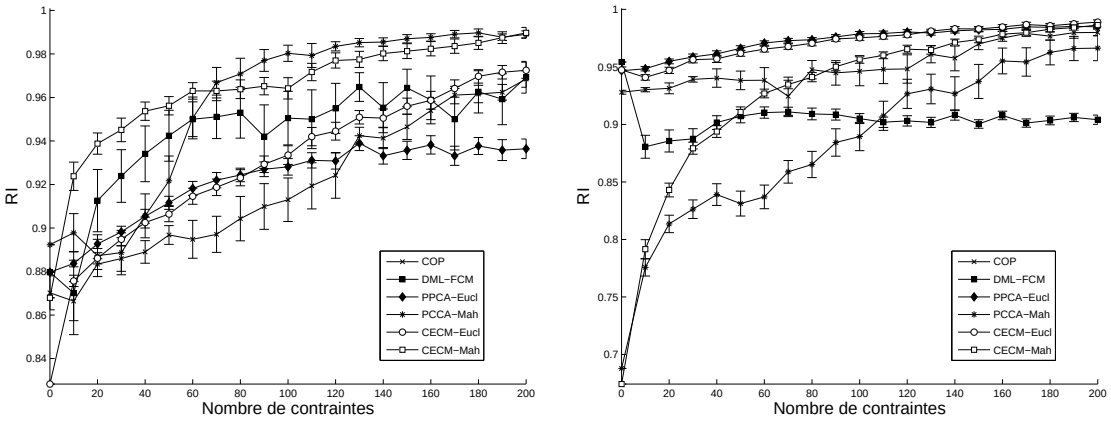
3.5.3 Comparaison des méthodes

La comparaison des performances avec des méthodes de la littérature est effectuée sur les jeux de données de l'UCI. Pour chaque jeu de données, six algorithmes sont comparés en utilisant un nombre de contraintes variant de 0 à 200. Ces algorithmes, présentés dans un tableau comparatif à l'annexe A, sont CECM et PCCA avec une distance de Mahalanobis et une distance Euclidienne (CECM-Mah, CECM-Eucl, PCCA-Mah, PCCA-Eucl), COP et DML suivi de FCM (noté DML-FCM sur les figures suivantes).

L'algorithme CECM utilise les mêmes paramétrages que dans la partie 3.4.1 et la version DML-FCM utilise une matrice pleine. Pour l'algorithme PCCA, nous avons choisi de fixer le nombre de classes à sa véritable valeur pour ainsi supprimer le terme pénalisant un nombre de classes trop élevé. De cette manière, PCCA correspond à CECM avec des valeurs des hyper-paramètres α et ρ élevées. La principale différence entre les deux algorithmes se situe alors dans leurs modes d'optimisation. En effet, la convergence de l'algorithme PCCA n'est pas toujours assurée dans le cas d'un nombre important de contraintes : la correction à effectuer due à l'omission de la contrainte (1.1) peut être trop importante et peut impliquer une augmentation de la valeur de la fonction objectif.

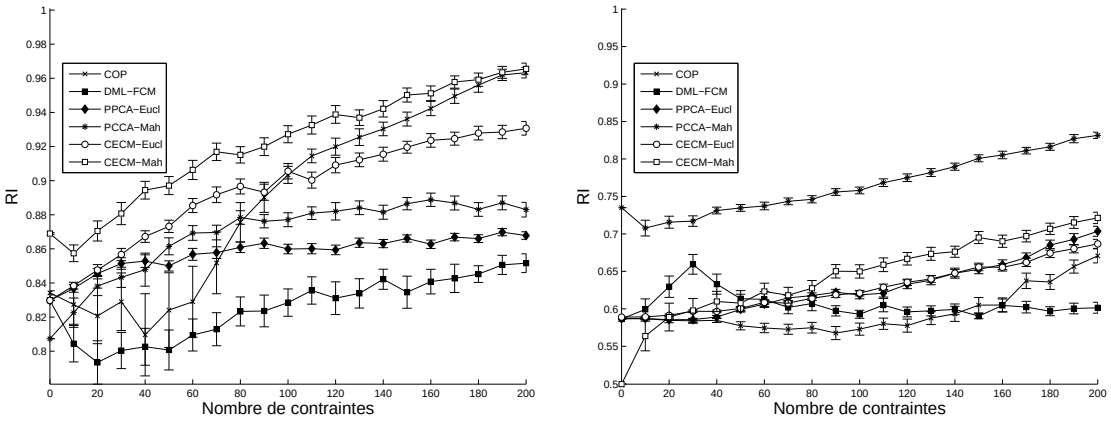
Les résultats sont présentés aux figures 3.24, 3.25 et 3.26. Chacune de ces figures montre l'indice de Rand moyen calculé sur 100 expériences avec une sélection aléatoire des contraintes. Il faut noter que comme COP ne trouve pas de solution faisable à chaque fois, notamment pour un nombre de contraintes élevé, les graphes montrent pour cet algorithme les 100 premières solutions faisables trouvées.

Dans un premier temps, nous pouvons observer que les meilleurs résultats sont obtenus avec CECM et PCCA. Pour le jeu de données Glass, CECM avec une distance de Mahalanobis donne de meilleurs résultats que PCCA quel que soit le nombre de contraintes. Pour Iris et LettersIJL avec une distance de Mahalanobis pour ces deux algorithmes, l'indice de Rand est meilleur pour PCCA quand le nombre de contraintes est élevé. Cependant il est intéressant de noter que lorsqu'il existe peu de contraintes, CECM donne de meilleurs résultats. Ceci est une caractéristique intéressante de l'algorithme car obtenir un nombre important de contraintes peut s'avérer difficile ou coûteux. Pour la base de données Wine, la distance Euclidienne est la distance la plus appropriée et les deux algorithmes PPCA et CECM fournissent des résultats similaires.



(a) (b)

Figure 3.24 – Comparaison des indices de Rand moyens et intervalles de confiance à 95% sur 100 essais pour CECM et différents autres algorithmes de classification par contraintes pour les jeux de données Iris et Wine.



(a) (b)

Figure 3.25 – Comparaison des indices de Rand moyens et intervalles de confiance à 95% sur 100 essais pour CECM et différents autres algorithmes de classification par contraintes pour les jeux de données Glass et Ionosphere.

Pour le jeu de données Ionosphere, la méthode PCCA avec une distance de Mahalanobis fonctionne mieux que CECM avec une distance identique, et ce, même sans contrainte. Ceci est dû à un problème de minimum local lié à l'initialisation par FCM des centres de gravité de CECM.

3.5.4 Complexité

Comme expliqué dans l'article de Masson et Denœux [44], le nombre de paramètres définissant une partition crédale est exponentiel par rapport au nombre de classes et linéaire par rapport au nombre d'objets. À chaque étape de l'algorithme CECM, un problème quadratique doit être résolu pour mettre à jour les fonctions de masses, ainsi qu'un système linéaire pour les centres de gravité. Par conséquent, l'approche reste limitée aux applications de taille modérée (c'est-à-dire aux applications de moins de 7 classes et de plusieurs centaines d'objets).

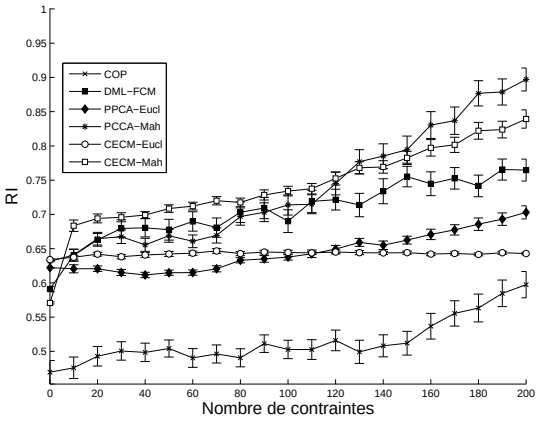


Figure 3.26 – Comparaison des indices de Rand moyens et intervalles de confiance à 95% sur 100 essais pour CECM et différents autres algorithmes de classification par contraintes pour le jeu de données LettersIJL.

Quelques comparaisons expérimentales ont été effectuées entre CECM et PCCA, les deux algorithmes présentant de bons résultats de classification dans la partie précédente. Tous les algorithmes ont été implémentés sous Matlab et ont été exécutés sur un ordinateur composé d'un processeur Dual Core AMD Opteron 885 et de 32 GO de RAM. Les tests sont effectués sur les jeux de données de l'UCI en utilisant 100 contraintes. Les tableaux 3.10 et 3.11 présentent le temps CPU et le nombre d'itérations moyennés sur 20 essais. L'algorithme CECM est souvent plus rapide que PCCA car le nombre d'itérations à effectuer pour trouver une solution est plus faible (cf. tableau 3.11).

Data	PCCA-Eucl.	PCCA-Mah	CECM-Eucl	CECM-Mah
Iris	5.2 ± 1.31	6.28 ± 1.78	2.74 ± 0.61	5.26 ± 1.17
Wine	6.47 ± 1.37	15.55 ± 4.91	4.36 ± 0.51	9.75 ± 2.73
Glass	0.69 ± 0.05	0.99 ± 0.08	2.5 ± 0.32	5.68 ± 1.98
Ionosphere	4.09 ± 4.11	42.30 ± 29.91	30.54 ± 5.30	56.03 ± 7.98
Letters	79.4 ± 45.24	63.07 ± 27.58	82.11 ± 29.42	37.46 ± 10.5

Tableau 3.10 – Moyenne et écart type du temps CPU (en secondes) sur 20 essais pour 100 contraintes choisies aléatoiremement.

Data	PCCA-Eucl.	PCCA-Mah	CECM-Eucl	CECM-Mah
Iris	25.23 ± 5.55	28.34 ± 8.02	5.00 ± 2.00	10.20 ± 3.02
Wine	26.10 ± 3.52	54.54 ± 15.87	5.40 ± 1.10	12.95 ± 4.71
Glass	42.30 ± 3.23	48.76 ± 4.31	5.65 ± 1.35	16.00 ± 8.01
Ionosphere	146.60 ± 148.79	75.75 ± 54.47	4.75 ± 0.55	9.30 ± 1.66
Letters	212.41 ± 122.13	161.30 ± 70.83	64.10 ± 25.05	29.85 ± 10.48

Tableau 3.11 – Moyenne et écart type du nombre d'itérations sur 20 essais pour 100 contraintes choisies aléatoiremement.

De plus, comme pour ECM, il est possible de réduire la complexité de CECM en considérant uniquement les sous-ensembles dont la cardinalité est faible. Par exemple, il est possible de restreindre les éléments focaux aux ensembles Ω, $\emptyset$ et aux sous-ensembles composés au plus de deux classes. De cette manière, le nombre de paramètres auparavant égal à $2^c n$ devient $c^2 n$. Ainsi, il existe un compromis acceptable entre la flexibilité de la méthode et son temps d'exécution. Pour illustrer ce principe, le jeu de données ToysDataVert est repris en ne considérant maintenant que quatre classes doivent être trouvées (ces classes correspondent aux quatre Gaussiennes). La version complète de CECM, nommée CECM-1 et qui consiste à utiliser pour chaque objet les 2^4 éléments focaux disponibles, est comparée à la version limitée de CECM, nommée CECM-2 et qui utilise pour chaque objet uniquement les sous-ensembles de cardinalité inférieure à 2 et les sous-ensembles $\emptyset$ et Ω. Les deux versions introduisent 100 contraintes choisies aléatoirement et donnent un indice de Rand de 0.99. Les temps CPU et le nombre d'itérations, moyennés sur 20 expériences, sont présentés au tableau 3.12. Nous pouvons ainsi observer qu'il existe une réduction significative du temps de calcul pour la version limitée, sans pour autant sacrifier la qualité de la classification. Par ce moyen, il est alors maintenant possible d'utiliser CECM dans le cadre d'applications d'une dizaine de classes.

	CECM-1 Eucl.	CECM-2 Eucl.	CECM-1 Mah.	CECM-2 Mah.
CPU(s)	141.27 ± 52.08	70.14 ± 25.52	144.79 ± 41.20	76.03 ± 29.15
Nb. Ité.	27.85 ± 11.65	22.00 ± 9.15	21.95 ± 7.16	21.05 ± 9.02

Tableau 3.12 – Comparaison entre la version complète (CECM-1) et la version limitée (CECM-2) pour l'algorithme CECM appliqué sur le jeu de données ToysDataVert avec $c = 4$. La moyenne et l'écart type pour le temps CPU et le nombre d'itérations sont calculés sur 20 essais pour 100 contraintes choisies aléatoirement.

Synthèse du chapitre

Dans ce chapitre, nous avons proposé une variante de l'algorithme évidentiel de classification automatique ECM qui consiste à utiliser une distance de Mahalanobis dans le but de gérer des classes de formes non sphériques. Une nouvelle méthode, dérivée de ECM et nommée CECM, est ensuite présentée. Elle permet de prendre en compte des connaissances a priori disponibles sous forme de contraintes sur des paires d'objets à classer.

Les expériences effectuées montrent que l'introduction de contraintes permet d'améliorer la qualité de la partition obtenue. Quand des modèles plus complexes sont utilisés, comme par exemple avec une métrique adaptative, les contraintes permettent d'obtenir une estimation des paramètres qui correspond mieux au problème considéré. L'algorithme CECM, qui permet de clarifier les ambiguïtés existantes, est finalement testé sur deux applications de segmentation d'images.

Une étude approfondie du comportement de l'algorithme a ensuite permis de proposer une nouvelle procédure d'apprentissage actif. Il en ressort que le nombre de

contraintes requis pour obtenir une classification correcte des données ne nécessite pas d'être très élevé. Enfin, les performances de CECM sont comparées avec plusieurs autres algorithmes de la littérature. Dans la plupart des cas, CECM donne les meilleurs résultats, surtout pour un nombre peu important de contraintes.

EVCLUS avec contraintes

Nous présentons maintenant un second algorithme de classification prenant en compte des contraintes Must-Link et Cannot-Link entre des paires d'objets. Cette nouvelle méthode, nommée CEVCLUS, est basée sur l'algorithme de classification évidentiel EVCLUS. Afin de constater l'amélioration des performances grâce à l'ajout de contraintes, CEVCLUS est évalué sur plusieurs jeux de données.

4.1 EVCLUS avec intégration de connaissance a priori

4.1.1 Expressions des contraintes et de la fonction objectif

Comme pour l'algorithme CECM, des éléments de connaissance a priori peuvent être intégrés à l'algorithme EVCLUS sous forme de contraintes entre des paires d'objets. Ces contraintes peuvent être exprimées en utilisant les plausibilités conjointes $pl_{i\times j}(\theta)$ et $pl_{i\times j}(\overline{\theta})$ (cf. paragraphe 3.2.1). Rappelons qu'une contrainte Must-Link correspond à une faible valeur de $pl_{i\times j}(\overline{\theta})$ et à une forte valeur de $pl_{i\times j}(\theta)$. Inversement, une contrainte Cannot-Link se traduit par une faible valeur de $pl_{i\times j}(\theta)$ et une forte valeur de $pl_{i\times j}(\overline{\theta})$.

De la même manière que pour l'algorithme ECM, nous proposons d'intégrer ces contraintes à EVCLUS en ajoutant un terme de pénalisation à la fonction objectif. Cependant le terme $J_{CONST}(\mathbf{M})$ de l'algorithme CECM (cf. équation 3.25) ne peut être utilisé tel quel : les objets atypiques ne sont en effet pas gérés de la même manière par les deux méthodes. L'algorithme ECM utilise l'hyperparamètre ρ, qui permet de réduire l'importance donnée à l'ensemble vide par rapport aux autres sous-ensembles. Cet hyperparamètre peut être vu comme une distance fixe entre les objets et le centre de gravité de la classe des objets considérés comme atypiques. L'algorithme EVCLUS, quant à lui, ne repose sur aucun modèle géométrique. Les objets atypiques ne sont donc pas définis par une certaine distance à un centre de gravité, mais en évaluant l'ensemble des distances de chaque paire d'objets du jeu de données. Si à présent le terme $J_{CONST}(\mathbf{M})$ est ajouté à l'algorithme EVCLUS, ce dernier deviendra trop sensible à l'ensemble vide, c'est-à-dire que le problème d'optimisation admettra plus facilement une masse élevée sur l'ensemble vide pour les objets contraints. Un nouveau terme de pénalisation doit donc être formulé.

En exploitant $pl_{i\times j}(\theta)$ et $pl_{i\times j}(\overline{\theta})$ pour une même contrainte, il est possible de définir $J_{\mathcal{M}}$ le coût de violer une contrainte Must-Link entre les objets i et j et $J_{\mathcal{C}}$ le

coût de violer une contrainte Cannot-Link pour ces mêmes objets :

$$J_{\mathcal{M}}(\mathbf{m}_i, \mathbf{m}_j) = pl_{i\times j}(\overline{\theta}) + 1 - pl_{i\times j}(\theta), \tag{4.1}$$
$$J_{\mathcal{C}}(\mathbf{m}_i, \mathbf{m}_j) = pl_{i\times j}(\theta) + 1 - pl_{i\times j}(\overline{\theta}). \tag{4.2}$$

Le nouvel algorithme, nommé Constrained EVCLUS (ou CEVCLUS), doit trouver une partition crédale compatible avec la matrice de dissimilarité donnée en entrée tout en respectant le plus possible les contraintes. Ainsi, la nouvelle fonction objectif à minimiser est la suivante :

$$J_{CEVCLUS}(\mathbf{M}, a, b) = J_{EVCLUS}(\mathbf{M}, a, b) + \xi \frac{1}{2(|\mathcal{M}| + |\mathcal{C}|)} J_{CONST_2}(\mathbf{M}), \tag{4.3}$$

avec

$$J_{CONST_2}(\mathbf{M}) = \sum_{(o_i,o_j)\in\mathcal{M}} J_{\mathcal{M}}(\mathbf{m}_i, \mathbf{m}_j) + \sum_{(o_i,o_j)\in\mathcal{C}} J_{\mathcal{C}}(\mathbf{m}_i, \mathbf{m}_j), \tag{4.4}$$

sous les contraintes (2.17). Le paramètre $\xi \in \mathbb{R}^+$ contrôle le compromis entre le modèle évidentiel et les contraintes.

4.1.2 Optimisation

Comme l'algorithme EVCLUS, les fonctions de masses peuvent être recodées de manière à satisfaire implicitement les contraintes de positivité et de sommation à 1 (cf. équation (2.17)), c'est-à-dire en employant (2.26). Nous pouvons alors minimiser la fonction objectif en utilisant une procédure de descente de gradient.

Mise à jour des coefficients a et b

Le terme de pénalisation J_{CONST_2} ne dépend pas des coefficients a et b. Les dérivées partielles de $J_{CEVCLUS}$ par rapport à ces coefficients sont donc identiques aux dérivées partielles de J_{EVCLUS} :

$$\frac{\partial J_{CEVCLUS}}{\partial a} = \frac{2}{\sum_{i<j} d_{ij}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{K_{ij}(aK_{ij} + b - d_{ij})}{d_{ij}}, \tag{4.5}$$

$$\frac{\partial J_{CEVCLUS}}{\partial b} = \frac{2}{\sum_{i<j} d_{ij}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{(aK_{ij} + b - d_{ij})}{d_{ij}}. \tag{4.6}$$

Mise à jour de la partition crédale M

Les masses étant reparamétrées conformément à (2.26), il est nécessaire de calculer la dérivée partielle de $J_{CEVCLUS}$ par rapport à α_{il} (où i est l'indice d'un objet et A_l celui d'un sous-ensemble de Ω) :

$$\frac{\partial J_{CEVCLUS}}{\partial \alpha_{il}} = \frac{\partial J_{EVCLUS}}{\partial \alpha_{il}} + \xi \frac{1}{2(|\mathcal{M}| + |\mathcal{C}|)} \frac{\partial J_{CONST_2}}{\partial \alpha_{il}}. \tag{4.7}$$

La dérivée de $J_{CEVCLUS}$ par rapport à α_{il} peut être reformulée comme suit (les détails sont fournis en annexe C) :

$$\frac{\partial J_{EVCLUS}}{\partial \alpha_{il}} = \frac{2a}{\sum_{i<j} d_{ij}} \sum_{j=i+1}^{n} \frac{(aK_{ij} + b - d_{ij})}{d_{ij}} \sum_{k,k'} \frac{\partial m_{ik}}{\partial \alpha_{il}} m_{jk'} s_{kk'}, \tag{4.8}$$

avec

$$\frac{\partial m_{ik}}{\partial \alpha_{il}} = \begin{cases} m_{ik}(1 - m_{ik}) & \text{si } l = k, \\ -m_{ik} m_{il} & \text{sinon.} \end{cases} \tag{4.9}$$

et

$$s_{kk'} = \begin{cases} 1 & \text{si } A_k \cap A_{k'} = \emptyset, \\ 0 & \text{sinon.} \end{cases} \tag{4.10}$$

Enfin, la dérivée du terme de pénalisation J_{CONST_2} par rapport à α_{il} est :

$$\frac{\partial J_{CONST_2}}{\partial \alpha_{il}} = \left(\sum_{(\mathbf{o}_i, \mathbf{o}_j) \in \mathcal{M}} \frac{\partial pl(\overline{\theta})}{\alpha_{il}} - \frac{\partial pl(\theta)}{\alpha_{il}} \right) + \left(\sum_{(\mathbf{o}_i, \mathbf{o}_j) \in \mathcal{C}} \frac{\partial pl(\theta)}{\alpha_{il}} - \frac{\partial pl(\overline{\theta})}{\alpha_{il}} \right) \tag{4.11}$$

$$\frac{\partial pl(\overline{\theta})}{\alpha_{il}} = -\frac{\partial m_{i\emptyset}}{\partial \alpha_{il}} + \frac{\partial m_{i\emptyset}}{\partial \alpha_{il}} m_{j\emptyset} - \sum_{A_k, |A_k|=1} \frac{\partial m_{ik}}{\partial \alpha_{il}} m_{jk}, \tag{4.12}$$

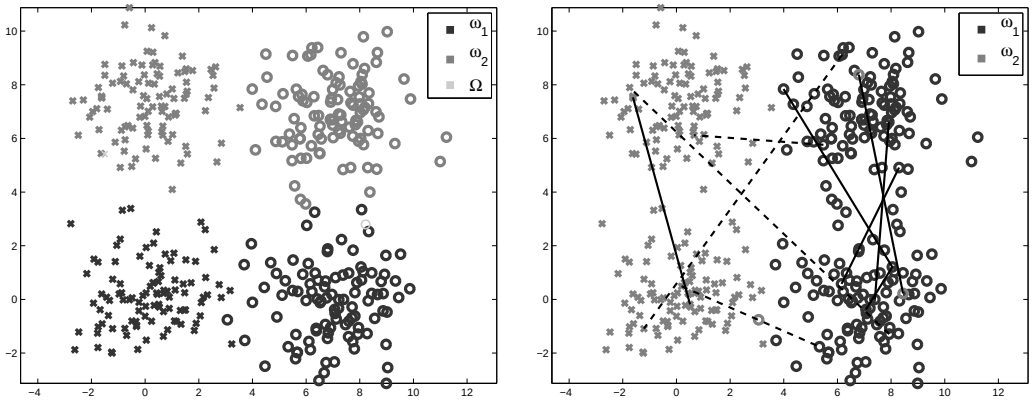
$$\frac{\partial pl(\theta)}{\alpha_{il}} = \sum_{A_k \cap A_{k'} \neq \emptyset} m_{jk'} \frac{\partial m_{ik}}{\partial \alpha_{il}}. \tag{4.13}$$

La méthode d'optimisation de la fonction objectif est une méthode de descente de gradient dont les détails techniques sont présentés dans [21]. Elle nécessite une initialisation des masses et des coefficients a et b et peut donner une solution correspondant à un minimum local. L'algorithme CEVCLUS doit donc être lancé plusieurs fois afin d'éviter ces minima locaux. Notons que la méthode d'optimisation utilisée comprend elle aussi des paramètres à fixer tels que le pas du gradient, le seuil du critère d'arrêt, etc. Pour les expériences suivantes, les mêmes paramétrages que l'article [21] sont utilisés.

4.1.3 Exemples illustratifs

Le jeu de données ToysDataVert (cf. paragraphe 3.1.3) est repris afin de montrer comment l'algorithme CEVCLUS peut être orienté vers une solution désirée en utilisant des contraintes appropriées. Sur ce jeu de données, l'algorithme EVCLUS donne une partition crédale avec une frontière horizontale ou verticale selon l'initialisation des masses. Le cas d'une frontière horizontale correspond à la solution la moins coûteuse en terme de fonction objectif (cf. figure 4.1(a)). Ce choix ne correspond cependant pas à la partition réelle du jeu de données. La figure 4.1(b) montre qu'ajouter aléatoirement 10 contraintes permet de guider l'algorithme vers la solution désirée. Notons que pour obtenir ce résultat, ξ est fixé à 1 et l'expérience est exécutée plusieurs fois. La solution ayant la fonction objectif minimale est ensuite

retenue. Il est possible de remarquer que cette partition pourrait être encore améliorée : deux objets contraints sont en effet mal classés. La solution finale correspond donc ici à un minimum local.



(a) (b)

Figure 4.1 – Partitions crédales nettes obtenues pour ToysDataVert en utilisant EVCLUS (a) et CEVCLUS avec 10 contraintes (b). Les symboles représentent les classes réelles et les lignes continues et en pointillés représentent les contraintes Must-Link et Cannot-Link.

L'algorithme CECM est très performant pour des jeux de données comprenant des classes de forme ellipsoïdale puisqu'il utilise une distance de Mahalanobis adaptée à chaque classe. À l'inverse, nous pouvons montrer que CEVCLUS, qui ne repose sur aucun modèle géométrique, peut gérer des classes non-linéairement séparables. Pour illustrer ce point, un nouveau jeu de données nommé ToysBananas est créé (cf. figure 4.2). Il est construit à partir de deux gaussiennes qui sont ensuite transformées à l'aide de l'équation cartésienne d'une ellipse. Chaque classe a alors une forme non hypersphérique ou non hyperellipsoïdale.

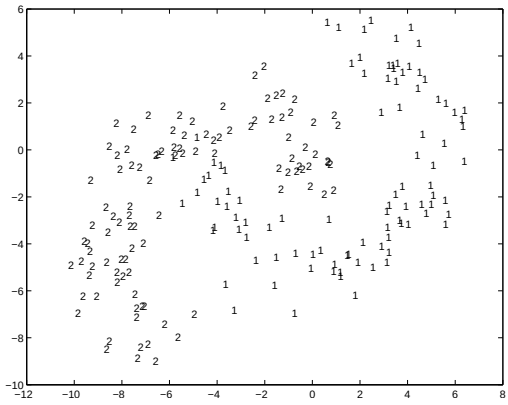


Figure 4.2 – Jeu de données ToysBananas généré automatiquement à l'aide de deux gaussiennes.

Comme le montre la figure 4.3(a), l'algorithme EVCLUS ne détecte pas correctement la forme des classes. Sans information supplémentaire, l'algorithme tend en effet vers des solutions pour lesquelles les classes sont sphériques ou ellipsoïdales.

L'ajout de 20 contraintes choisies aléatoirement permet à l'algorithme de guider l'algorithme vers une solution différente (cf. figure 4.3 (b)).

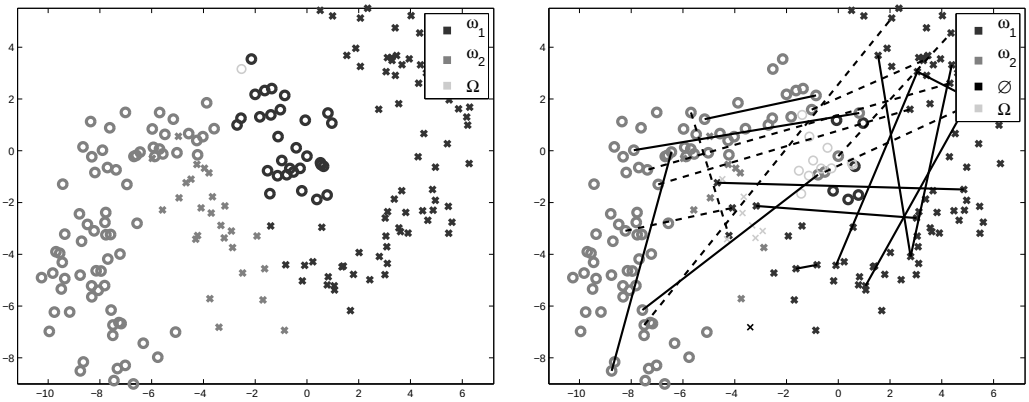


Figure 4.3 – Partitions crédales nettes pour le jeu de données ToysBananas obtenues avec EVCLUS (a) et CEVCLUS avec $\xi = 1$ (b). Les symboles représentent les classes réelles des objets et les lignes continues et en pointillés correspondent aux contraintes Must-Link et Cannot-Link.

4.2 Protocole expérimental

Afin d'évaluer les performances de l'algorithme CEVCLUS, un protocole semblable à celui de l'algorithme CECM a été mis en place. Dans un premier temps, nous présentons les différents jeux de données utilisés, puis nous expliquons comment la qualité de la partition trouvée par CEVCLUS peut être évaluée.

4.2.1 Données

Données de l'UCI

Nous avons utilisé les jeux de données de l'UCI présentés à la partie 3.3.1. Ces données étant sous une forme vectorielle et CEVCLUS n'acceptant en entrée que des données relationnelles, une matrice de dissimilarité basée sur le calcul des distances entre objets doit être construite. Nous supposons qu'il n'existe pas de moyen de connaître la distance la plus appropriée pour ces jeux de données. Par conséquent, la distance Euclidienne est choisie par défaut.

La figure 4.4 représente les matrices de dissimilarité pour les différents jeux de données de l'UCI. Comme les classes réelles sont connues, l'ordre de présentation des objets est établi en regroupant les objets par classe puis en utilisant la méthode de réorganisation VAT [5] (cf. paragraphe 1.1.1). Les limites des classes sont affichées par des lignes noires. De cette manière, les blocs diagonaux représentent les individus de la même classe. Ils sont donc susceptibles d'avoir une faible dissimilarité et donc d'être représentés par des niveaux de gris foncé. À l'inverse, l'extérieur des blocs représente la dissimilarité entre des objets de classe différentes. Ils sont donc supposés avoir une dissimilarité forte, c'est-à-dire qu'ils sont représentés par pixels ayant des niveaux de gris clair. Par la suite, nous utiliserons exclusivement ce procédé de

séparation par classe suivi de VAT pour représenter les matrices de dissimilarité. Il est possible d'observer à l'aide de ces figures qu'il existe pour Iris deux classes proches l'une de l'autre et une troisième classe plus éloignée (cf. figure 4.4(a)). Nous pouvons aussi remarquer que les classes de Wine sont naturellement bien séparées (cf. figure 4.4(b)). La distance Euclidienne est donc bien adaptée à ce jeu de données, contrairement au jeu de données LettersIJL où les classes sont très allongées (cf. figure 4.4(e)).

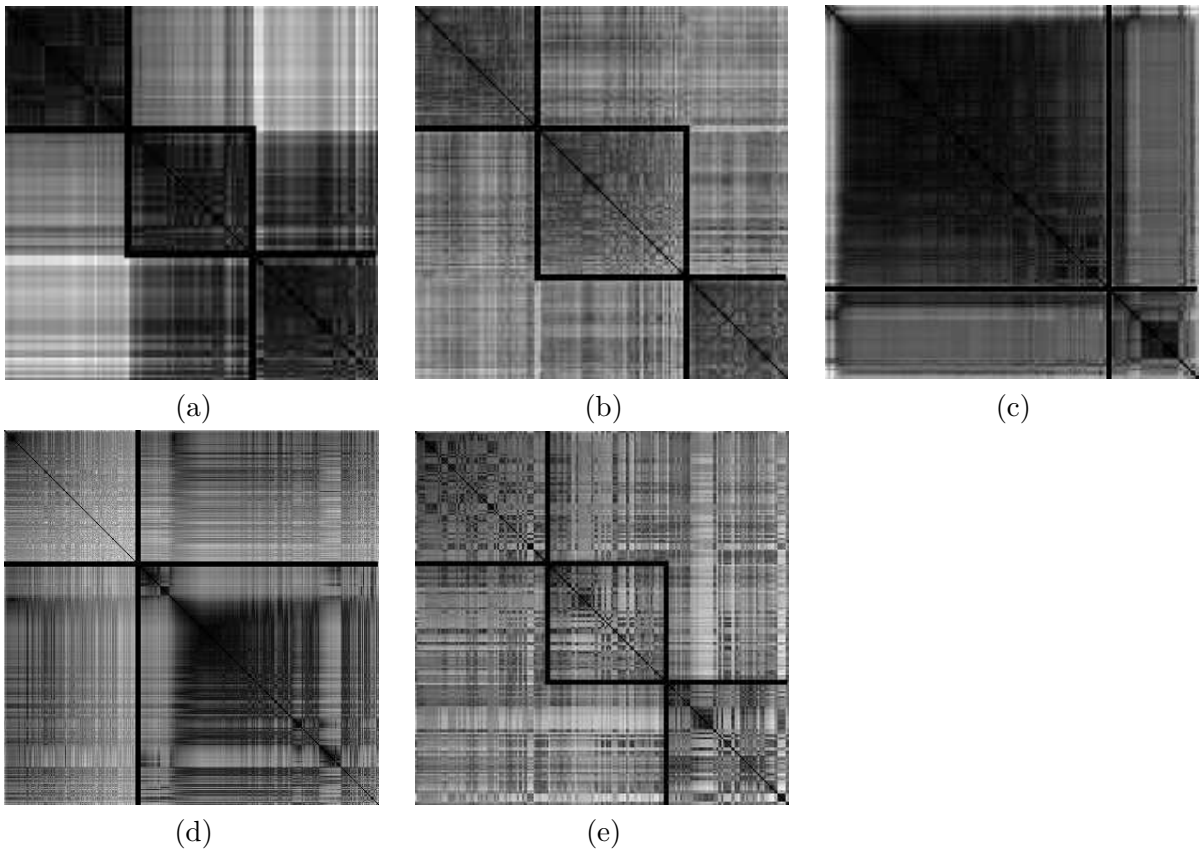


Figure 4.4 – Matrices de dissimilarité obtenues à l'aide de la distance Euclidienne pour Iris (a), Wine (b), Glass (c), Ionosphere (d) et LettersIJL (e). Les traits noirs démarquent les différentes classes réelles.

Données 20Newsgroups

Le jeu de données 20Newsgroups[1] est composé d'environ 20 000 messages récupérés à partir de 20 groupes de discussion. Ces groupes de discussion sont plus ou moins similaires et peuvent donc être classés selon différents thèmes :

- Le thème de l'informatique comprend les groupes de discussion comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware et comp.windows.x.
- Le thème du loisir correspond à un ensemble de sujets tels que le sport, ou les voitures. Il comprend les groupes de discussion rec.autos, rec.motorcycles, rec.sport.baseball et rec.sport.hockey.

1. Disponible sur le site `http://people.csail.mit.edu/jrennie/20Newsgroups/`.

- Le thème de la science est composé de messages parlant d'astronomie, de biologie et de physique. Ces documents se trouvent dans les forums de discussion sci.crypt, sci.electronics, sci.med et sci.space.
- Le thème du commerce est construit à partir d'un unique groupe de discussion : misc.forsale.
- Le thème de la politique correspond aux forums de discussion talk.politics.misc, talk.politics.guns et talk.politics.mideast.
- Enfin, le thème de la religion peut regrouper les forums de discussion talk.religion.misc, alt.atheism et soc.religion.christian.

Pour nos expériences, nous utilisons un échantillon de ces groupes de discussion, en conservant ceux dont les noms commencent par comp, rec, sci et talk. Quatre classes dont les sujets sont l'informatique, les loisirs, les sciences et les discussions controversés ont ainsi été formées. Les 16242 messages de ce nouveau jeu de données sont ensuite caractérisés par 100 mots : pour chaque message la présence d'un mot est indiquée par une valeur de 1 et l'absence d'un mot par une valeur de 0. Les données correspondent donc maintenant à des données vectorielles composées de 100 caractéristiques binaires. Nous sélectionnons aléatoirement parmi ces données 7% d'individus de chaque classe (cf. table 4.1).

classe	sujet	nb. objets
1	Informatique	322
2	Loisir	247
3	Sciences	186
4	Sujets controversés	382
	total	1136

Tableau 4.1 – Caractéristiques d'un échantillon du jeu de données 20Newsgroups.

Enfin, pour obtenir une matrice de dissimilarité, nous utilisons une mesure de distance basée sur la corrélation des objets [47] :

$$D_{corr}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{2}\left(1 - \frac{{\mathbf{x}_1}^\top \mathbf{x}_2}{\|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|^2 - 2{\mathbf{x}_1}^\top \mathbf{x}_2}\right). \tag{4.14}$$

La matrice obtenue correspond à la figure 4.5. Nous pouvons observer que les classes sont difficilement reconnaissables. La dissimilarité entre objets de même classe a en effet souvent une valeur très élevée et certains objets de classes différentes sont très proches. Il existe effectivement des mots pouvant être considéré comme du bruit. C'est le cas par exemple du mot "mac" qui est censé apparaître uniquement dans une discussion sur le thème de l'informatique. Ce mot existe cependant dans un nombre important de messages issus des autres classes. En examinant plus attentivement les messages, il est possible de s'apercevoir que "mac" correspond aussi au nom d'une personne.

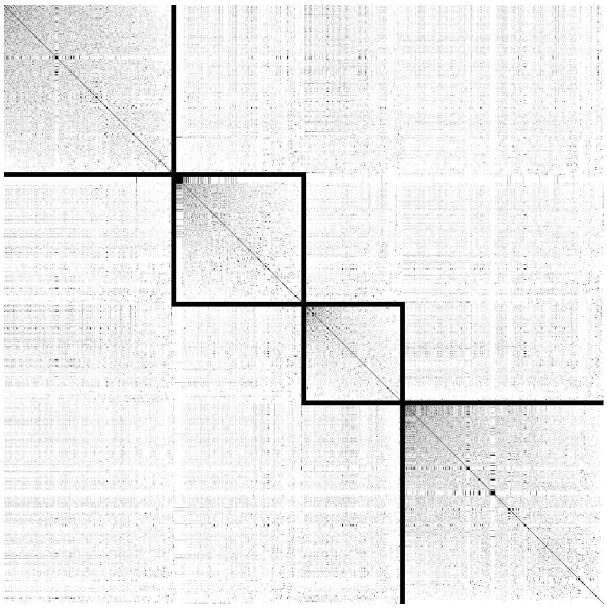


Figure 4.5 – Matrice de dissimilarité obtenues à l'aide d'une mesure de distance basée sur la corrélation des objets pour le jeu de données 20Newsgroups.

Données ChickenPieces

Le jeu de données chickenPieces [2] est composé de 446 images. Chaque image est constituée d'une silhouette en noir et blanc d'un morceau particulier de poulet (cf. figure 4.6). Le nombre de classes de ce jeu de données correspond au nombre de parties du poulet représentées. La distribution des classes est indiquée dans le tableau 4.2.

(a) (b) (c) (d) (e)

Figure 4.6 – Images provenant du jeu de données ChickenPieces. Chaque image représente une partie du poulet : le blanc (a), le dos (b), la cuisse et le pilon (c), l'aile (d) et le pilon (e).

Afin d'obtenir une matrice de dissimilarité, les images sont floutées puis une distance est calculée entre les contours des images prises deux à deux [10]. Comme différents paramétrages sont testés, 44 matrices de dissimilarité sont construites. Nous choisissons de retenir la matrice de dissimilarité $S = (s_{ij})$ nommée chickenPieces-20-90 [3]. Cette matrice étant légèrement asymétrique, une nouvelle matrice $D = (d_{ij})$ visible à la figure 4.7 est calculée comme suit : $d_{ij} = \frac{1}{2}(s_{ij} + s_{ji})$.

2. Disponible sous `http://www.iam.unibe.ch/fki/databases/string-edit-distance-matrices`
3. Disponible sous `http://prtools.org/disdatasets`.

classe	partie	nb. images
1	Blanc	96
2	Dos	76
3	Cuisse et pilon	61
4	aile	117
5	pilon	96
	total	446

Tableau 4.2 – Caractéristiques du jeu de données ChickenPieces.

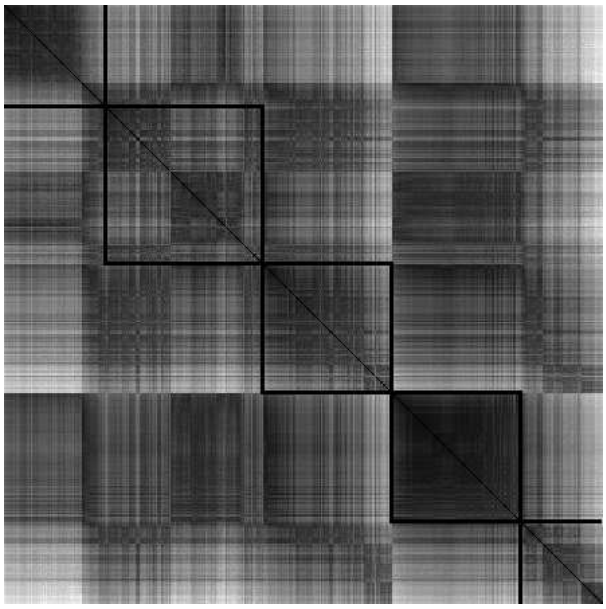


Figure 4.7 – Matrice de dissimilarité du jeu de données ChickenPieces.

4.2.2 Évaluation de la qualité de la partition

Les performances de CEVCLUS sont testées sur des bases de données dont les classes sont connues a priori. Il est donc possible d'utiliser l'indice de Rand (cf. équation (3.27)) de la même manière que pour CECM.

Rappelons que l'algorithme CEVCLUS peut converger vers un minimum local. De ce fait, pour déterminer une solution nous exécutons 10 fois l'algorithme à partir d'initialisations différentes : neuf initialisations aléatoires et une initialisation correspondant à la solution de EVCLUS. La solution retenue est celle qui a la fonction objectif de valeur minimale. Remarquons que pour certaines expériences, les contraintes sont initialisées aléatoirement. Dans ce cas, nous répétons la procédure décrite ci-dessous plusieurs fois, afin d'obtenir une moyenne et un intervalle de confiance à 95% (en supposant la distribution normale) de l'indice de Rand et du temps d'exécution.

L'hyperparamètre ξ, qui contrôle le degré d'importance donné aux contraintes, peut influencer grandement les résultats. Le comportement de l'algorithme en fonction de différentes valeurs de cet hyperparamètre est donc étudié dans la partie 4.4.1.

Enfin, le nombre de classes c étant toujours connu pour les expériences qui suivent, ce paramètre est fixé a priori. Cependant, il est toujours possible d'utiliser l'indice de validité de EVCLUS pour déterminer c (cf. paragraphe 2.2.3).

4.3 Intérêt de l'ajout de contraintes

Cette partie étudie l'intérêt d'ajouter des contraintes à un problème de classification donné. Des expériences sont tout d'abord réalisées sur les jeux de données de l'UCI. Un exemple d'application est ensuite présenté.

4.3.1 Ajout de contraintes

Données de l'UCI

Dans ces expérimentations, la sélection de contraintes est automatique. Elle consiste à sélectionner aléatoirement des paires d'objets puis à leur attribuer une contrainte Must-Link ou Cannot-Link en fonction de leurs classes réelles. Comme les résultats de classification peuvent être très sensibles aux choix des contraintes, 100 essais sont exécutés pour un nombre de contraintes fixé. Les figures 4.8, 4.9 et 4.10 montre l'indice de Rand moyen en fonction du nombre de contraintes. L'hyperparamètre ξ est fixé à 1.

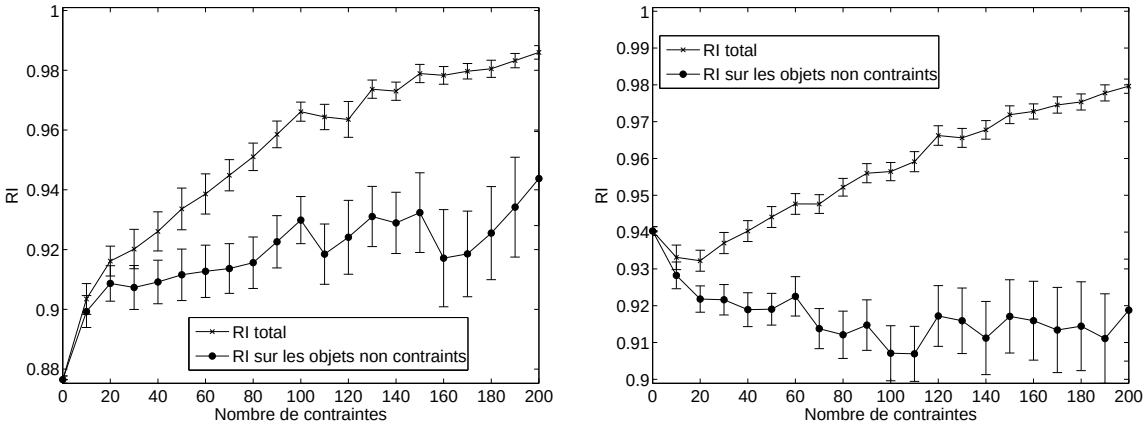


Figure 4.8 – Indices de Rand moyens et intervalles de confiance à 95% trouvés avec l'algorithme CEVCLUS tel que $\xi = 1$ pour Iris (a) et Wine (b).

Ces figures montrent que l'indice de Rand total augmente avec le nombre de contraintes. Sur trois jeux sur cinq, l'indice de Rand sur les objets non contraints est lui aussi meilleur que l'indice de Rand pour EVCLUS. Cela signifie qu'en général, ajouter des contraintes profite à l'ensemble des objets. Il faut noter de plus qu'à l'exception du jeu de données Wine, la distance n'est pas adaptée à la structure des données. Par exemple, dans le cas du jeu de données Glass, un nombre important de points sont caractérisés par une masse élevée allouée à l'ensemble vide. Les contraintes permettent donc de contrebalancer l'utilisation d'une métrique peu adaptée.

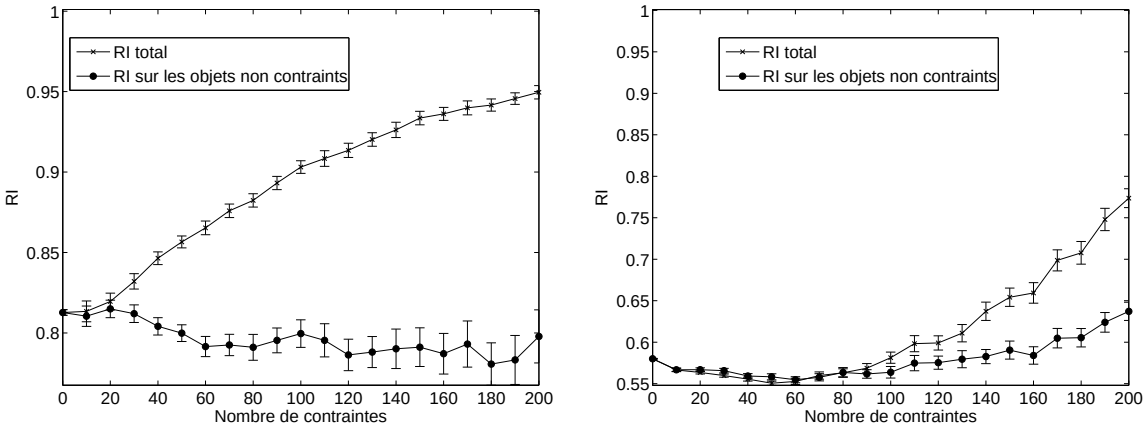


Figure 4.9 – Indices de Rand moyens et intervalles de confiance à 95% trouvés avec l'algorithme CEVCLUS tel que $\xi = 1$ pour Glass (a) et Ionosphere (b).

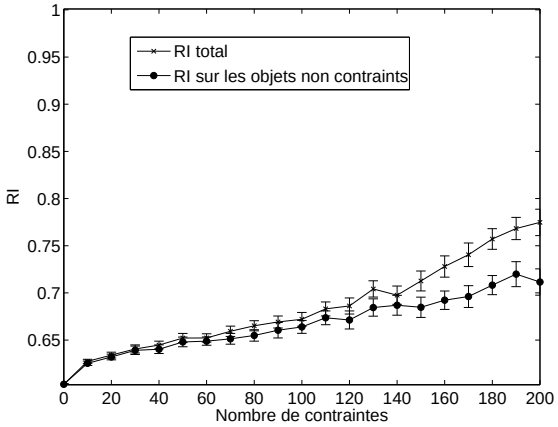


Figure 4.10 – Indices de Rand moyen et intervalles de confiance à 95% trouvés avec l'algorithme CEVCLUS tel que $\xi = 1$ pour LettersIJL.

Application

Le jeu de données 20Newsgroups est utilisé pour montrer une application possible de l'algorithme EVCLUS. Nous proposons en effet de construire à partir des données vectorielles de ce jeu des règles permettant de constituer un ensemble de contraintes Must-Link et Cannot-Link. Ainsi, tous les documents qui contiennent les mots "bible" et "religion" doivent être dans la même classe. Il en va de même pour les mots "computer" et "disk", "computer" et "win", "games" et "win", etc. De plus, ces couples de mots peuvent être utilisés pour créer des contraintes Cannot-Link : un message contenant les mots "bible" et "religion" n'est pas dans la même classe qu'un message utilisant les mots "computer" et "disk". Le schéma 4.11 présente les différents mots pris en compte. En tout, 3947 contraintes sont créées. Notons que parmi ces contraintes, certaines peuvent être fausses : après une étude des classes réelles des objets contraints, il est possible de constater qu'il existe 4.1% de contraintes bruitées.

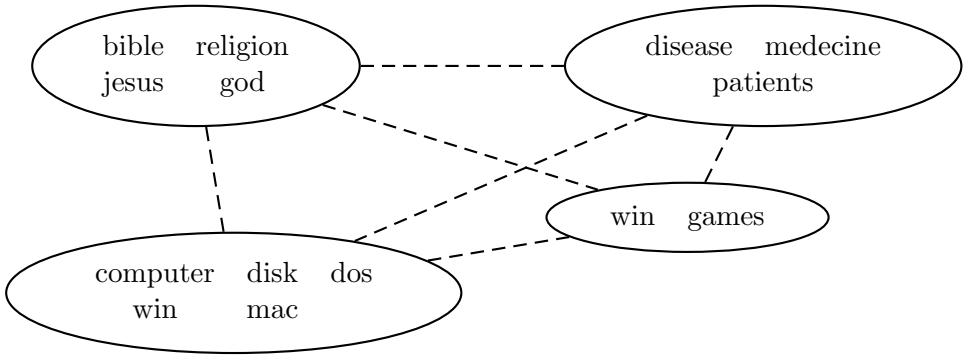


Figure 4.11 – Schéma des règles utilisées pour construire les contraintes Must-Link et Cannot-Link pour la base de données 20Newsgroups. Les cercles incluent des groupes de mots identiques. Les messages contenant une paire de mots située dans un même groupe sont contraints par un Must-Link, et les messages contenant au moins deux mots d'un groupe ont une contrainte Cannot-Link avec les messages contenant au moins deux mots issus d'un autre groupe.

L'algorithme EVCLUS est tout d'abord exécuté sans contraintes. La mesure de non-spécificité globale obtenue est de 0.52. Ainsi, la plupart des objets sont classés de manière incertaine. L'indice de Rand calculé à partir de la partition finale est de 0.65. Ce résultat peu performant n'est pas surprenant : la matrice de dissimilarité du jeu de données montre effectivement que de nombreux objets ayant une dissimilarité forte sont en réalité issus de la même classe. L'algorithme CEVCLUS est ensuite exécuté 10 fois en utilisant $\xi = 1$ et la solution qui a la plus faible valeur de fonction objectif est retenue. La mesure de non-spécificité globale de cette solution est de 0.27 et l'indice de Rand est de 0.70. L'ajout de contraintes a donc permis d'éliminer la plupart des incertitudes et d'améliorer le résultat de classification. Cette amélioration peut paraître faible par rapport au nombre de contraintes introduites, cependant il faut noter que la plupart des contraintes sont peu informatives. Par exemple, les messages contenant des mots religieux sont souvent déjà regroupés dans une même classe par EVCLUS.

Discussion

Il arrive parfois que la solution trouvée par EVCLUS soit meilleure que la solution trouvée par CEVCLUS. C'est le cas par exemple des jeux de données Wine (pour 10 à 30 contraintes, cf. figure 4.8(b)) et Ionosphere (pour 10 à 80 contraintes, cf. figure 4.9(b)). Le tableau 4.3 reprend ces expériences et indique le nombre de solutions trouvées qui sont moins intéressantes qu'une solution d'EVCLUS. De manière générale, plus il existe de contraintes et plus le nombre de solutions dégradées diminue. Enfin, selon les jeux de données, les chutes de performances peuvent être plus ou moins importantes pour un nombre réduit de contraintes.

Tout comme expliqué pour CECM dans le paragraphe 3.4.2, ces chutes de performances peuvent être dues à deux raisons. La première raison correspond à la situation suivante : l'ajout d'une contrainte peut avoir l'effet inverse de celui désiré : l'un des deux points contraints, mal classé de manière sûre (c'est-à-dire avec un fort degré de croyance), pousse le second, bien classé, dans la mauvaise classe.

Jeux de données \ Nb contraintes	10	20	30	40	50
Iris	12%	5%	7%	6%	6%
Wine	59%	75%	60%	54%	51%
Glass	38%	32%	13%	3%	0%
Ionosphere	95%	98%	97%	99%	99%
LettersIJL	2%	1%	1%	0%	2%

Tableau 4.3 – Pourcentage de chute de performances de CEVCLUS par rapport à EVCLUS.

L'algorithme peut alors entraîner d'autres points (contraints ou non) vers des classes autres que celles attendues (cf. figure 4.12(b)). Ainsi, CEVCLUS converge vers une solution non désirée. Le second comportement induisant une chute de performance est la concentration de points contraints dans une zone particulière de l'espace associé à la base de données. Ces points sont affectés de manière sûre à une classe et cela conduit à modifier fortement la région concernée, et par propagation l'ensemble de l'espace. Ainsi, la figure 4.13(b) montre que les points avoisinant les points contraints sont affectés à la même classe ω_1 et les objets qui se trouvent à la frontière opposée (entre ω_1 et ω_3) sont maintenant inclus dans la classe ω_3 alors qu'ils étaient autrefois dans la bonne classe, c'est-à-dire ω_1.

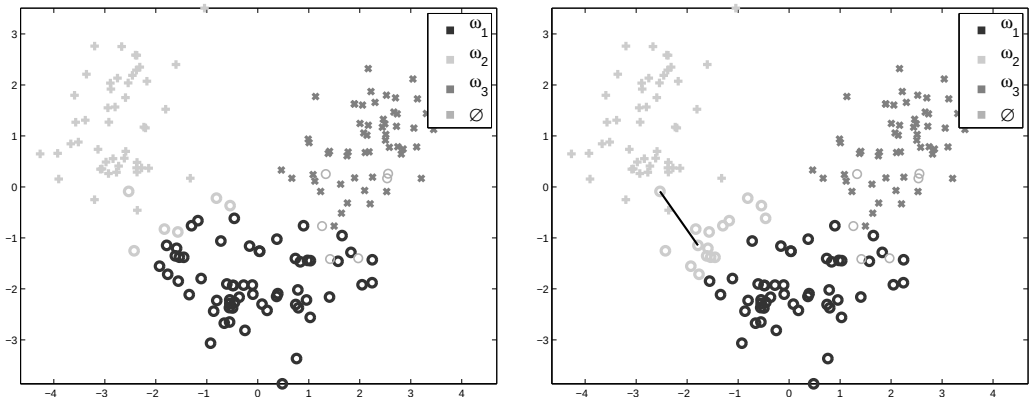


Figure 4.12 – Partitions crédales nettes de la base Wine obtenue par EVCLUS (a) et CEVCLUS (b). L'initialisation de CEVCLUS correspond à la solution trouvée par EVCLUS. L'algorithme CEVCLUS utilise ξ=1 et une contrainte Must-Link représentée par un segment de droite. Les symboles correspondent aux classes réelles des objets.

Le choix de la distance utilisée pour générer la matrice de dissimilarité a aussi un impact sur la qualité de la partition finale. Les contraintes intégrées dans l'algorithme EVCLUS peuvent aider à l'adaptation de cette distance (par exemple pour le jeu de données Iris) mais cela demande parfois un nombre important de contraintes avant d'obtenir un résultat conséquent (c'est le cas pour le jeu de données Ionosphere).

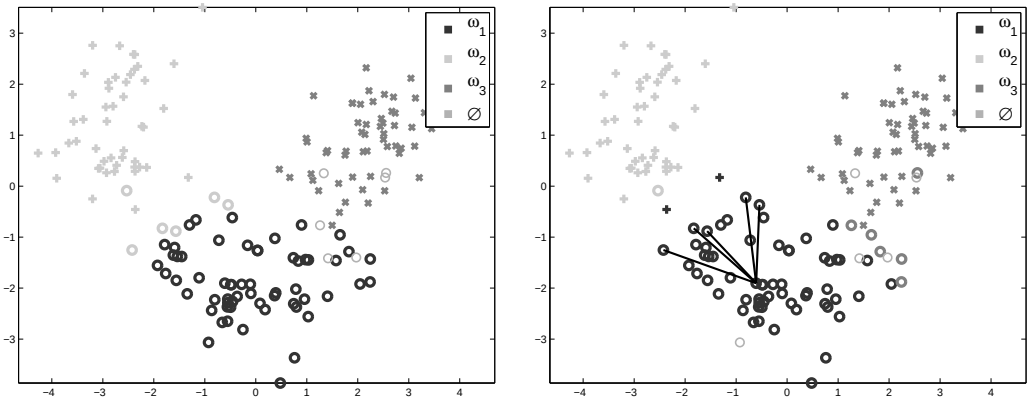


Figure 4.13 – Partitions crédales nettes de la base Wine obtenue par EVCLUS (a) et CEVCLUS (b). L'initialisation de CEVCLUS correspond à la solution trouvée par EVCLUS. L'algorithme CEVCLUS utilise cinq contraintes Must-Link situées dans une zone réduite de l'espace des données. Les contraintes sont représentés par des segments de droite. Les symboles correspondent aux classes réelles des objets.

4.3.2 Apprentissage actif

Rappelons que le principe de l'apprentissage actif consiste à choisir automatiquement des paires d'objets et à consulter un expert pour connaître la nature de leur relation (cf. partie 1.2.3). Comme ce procédé peut être coûteux en temps pour l'expert, notamment pour un nombre de contraintes élevé, il est important de sélectionner le moins possible de points tout en améliorant le plus possible la qualité de la partition. Dans le cadre de l'algorithme CECM, la stratégie de sélection d'une contrainte consistait à réunir un point très incertain avec un point très certain (cf. paragraphe 3.4.2). Les classes réelles des jeux de données utilisés dans ces expériences étant connues, le type de contrainte (Must-Link ou Cannot-Link) était automatiquement trouvés, sans l'aide d'un expert. Ce procédé nous a permis que comparer l'apprentissage actif avec le choix de contraintes aléatoire. Pour l'algorithme CEVCLUS, nous avons opté pour une stratégie d'apprentissage actif pouvant être utilisée dans le cadre d'une application réelle : pour un objet incertain, l'expert se voit proposer plusieurs objets certains issus des différentes classes existantes. De cette façon, l'utilisateur peut d'une part avoir une idée des caractéristiques de chaque classe et d'autre part déterminer avec certitude la nature d'au moins une contrainte.

La stratégie utilisée est donc la suivante : dans un premier temps, tout comme pour CECM, un premier objet très incertain est choisi. Le degré d'incertitude est ici mesuré par la non-spécificité. Un ensemble d'objets très certains est ensuite sélectionné, de manière à ce que chaque classe soit représentée par au moins un objet. Il y a donc au moins c nouveaux objets choisis. La certitude d'un objet ne pouvant plus être définie comme CECM par le centre de gravité des classes, c'est la mesure de non-spécificité qui est utilisée. Enfin, l'utilisateur peut choisir de contraindre l'objet incertain avec n'importe quel objet certain sélectionné automatiquement par l'algorithme.

La mise en place d'un apprentissage actif dans le cas d'un algorithme de classification utilisant des données relationnelles peut sembler délicat. Comment faire effectivement, quand il n'existe pas de description explicite des données (c'est-à-dire aucune caractéristique propre à chaque individu), pour présenter deux objets à un expert ? Pour cela, il semble nécessaire de revenir à la source de ce qui a construit les matrices de dissimilarités. Ainsi pour les données ChickenPieces, il est possible de présenter à l'expert les images d'origine qui ont été utilisées pour créer la matrice de dissimilarités. L'intérêt de l'apprentissage actif est donc montré à l'aide de cette base de données. L'algorithme EVCLUS est tout d'abord exécuté 10 fois et la solution de coût minimal est sélectionné. L'indice de Rand est alors de 0.76. Grâce à la partition crédale trouvée, la mesure de non-spécificité peut être calculée pour chaque point de la base de données. Ainsi, la stratégie d'apprentissage actif peut être mise place. La figure 4.14 montre les images correspondant aux différents objets sélectionnés par l'apprentissage actif. L'image centrale représente l'objet incertain et les huit images qui l'entourent les objets les plus certains de chaque classe trouvée. L'image encadrée correspond à l'objet certain le plus proche de l'objet incertain. Ceci permet d'indiquer à l'utilisateur quelle est la contrainte la plus importante à déterminer. Une fois que l'expert a fixé des contraintes (cf. figure 4.15(a)), l'algorithme CEVCLUS est exécuté en utilisant pour initialisation la solution de EVCLUS. Le nouvel indice de Rand obtenu est de 0.80. Les contraintes ont donc bien guidé l'algorithme vers une meilleure solution. La méthode de sélection de contraintes peut de nouveau être utilisée avec le résultat de CEVCLUS (cf. figure 4.15(b)), ce qui permet d'obtenir un indice de Rand de 0.82. Ce procédé est répété cinq fois et les derniers indices de Rand obtenus sont tous de 0.83. La figure 4.15(c) montre le troisième choix effectué par l'utilisateur.

4.4 Évaluation de la méthode

Dans la partie précédente, nous avons démontré l'intérêt de l'ajout de contraintes et avons présenté le comportement général de l'algorithme. Dans cette partie, nous nous intéressons à différents aspects de l'algorithme tels que le choix de ξ ou sa complexité afin d'approfondir nos connaissances sur son comportement et d'améliorer la qualité des résultats.

4.4.1 Choix de l'hyper-paramètre ξ

Le paramètre ξ permet de contrôler l'importance donnée aux contraintes par rapport au modèle évidentiel. Pour savoir comment fixer ce paramètre, plusieurs expériences sont exécutées sur les jeux de données de l'UCI. Ces expériences consistent à tester différentes valeurs de ξ pour un nombre de contraintes fixé. L'évolution de l'indice de Rand moyen (calculé sur 100 essais) en fonction de ξ est représentée par les tableaux 4.4 et 4.5.

Les résultats montrent que pour un faible nombre de contraintes, le meilleur paramétrage de ξ dépend surtout du jeu de données. En effet, pour Wine avec 20 et 50 contraintes, les meilleurs résultats sont trouvés lorsque $\xi = 0.05$ alors que pour Iris

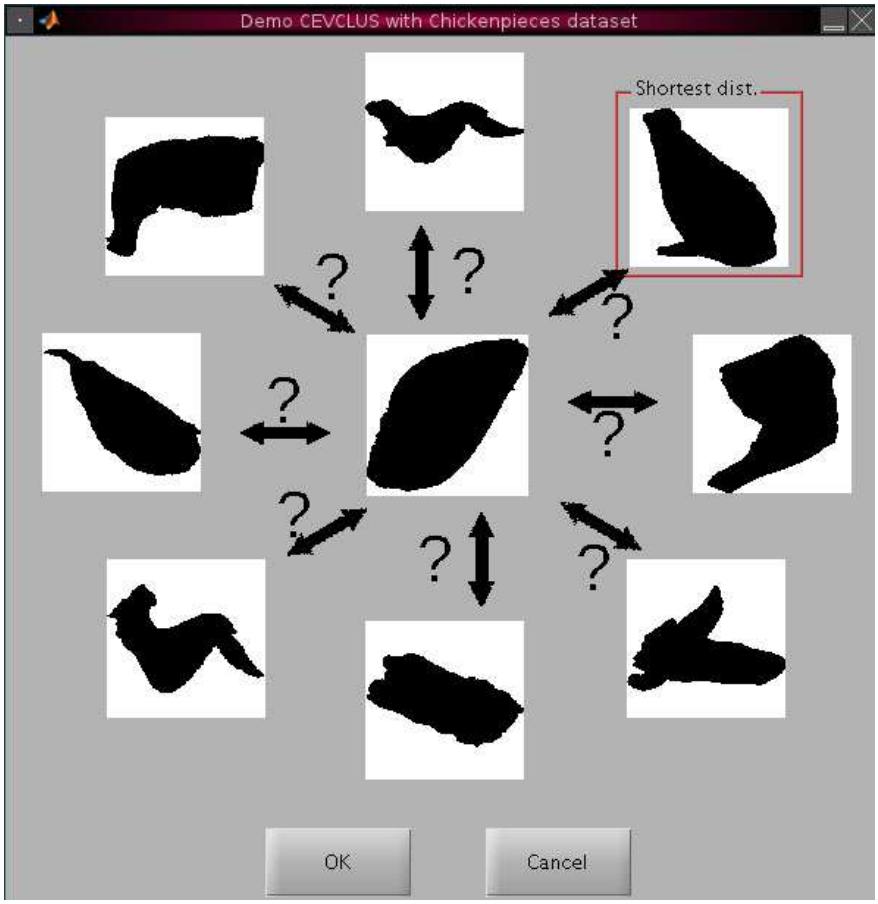
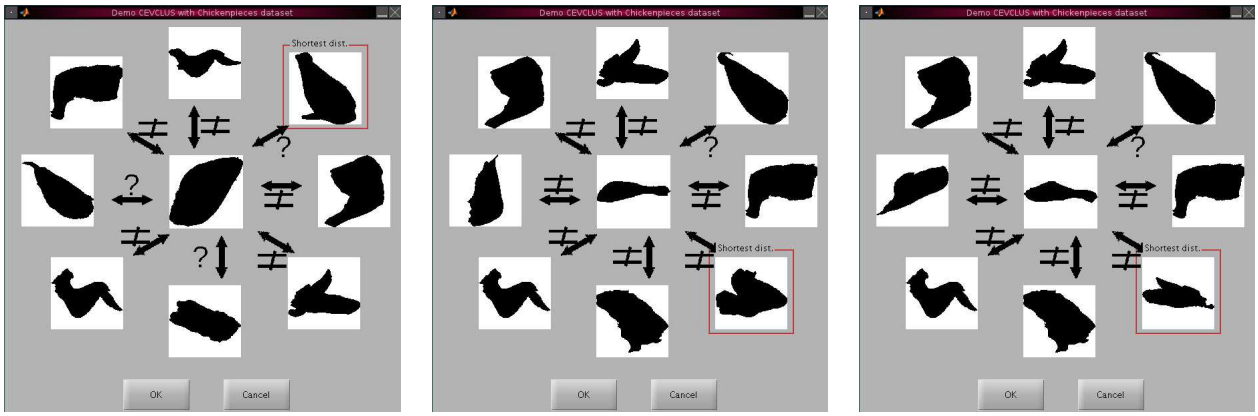


Figure 4.14 – Interface graphique pour la base de données ChickenPieces après exécution de l'algorithme EVCLUS. L'image centrale représente l'objet le moins certain et les cinq premières images qui l'entourent (en partant de l'objet situé au plus haut) les objets les plus certains de chaque classe trouvée par EVCLUS. Les trois objets restants sont les objets les plus certains, sans considération de classes. L'objet encadré correspond à l'objet le plus proche de l'objet incertain.



(a) (b) (c)

Figure 4.15 – Choix de l'utilisateur pour la première (a), deuxième (b) et troisième (c) itération de l'apprentissage actif. Le symbole "≠" représente une contrainte Cannot-Link alors que le symbole "=" une contrainte Must-Link. Le symbole "?" correspond à une incertitude de la part de l'expert. La contrainte n'est alors pas prise en compte.

C	ξ	Iris	Wine	Glass	Ionosphere	LettersIJL
20	0	0,88 ± 0,00	0,94 ± 0,00	0,81 ± 0,00	0.58 ± 0,00	0,60 ± 0,00
	0.05	0,88 ± 0,00	0,95 ± 0,00	0,78 ± 0,00	0,56 ± 0,00	0,63 ± 0,00
	0.1	0,88 ± 0,00	0,94 ± 0,00	0,79 ± 0,00	0,55 ± 0,00	0,64 ± 0,00
	0.2	0,87 ± 0,01	0,93 ± 0,00	0,82 ± 0,00	0,56 ± 0,00	0,64 ± 0,00
	0.5	0,90 ± 0,01	0,93 ± 0,00	0,83 ± 0,00	0,56 ± 0,00	0,63 ± 0,00
	0.8	0,91 ± 0,01	0,93 ± 0,00	0,83 ± 0,00	0,56 ± 0,00	0,63 ± 0,00
	1	0,91 ± 0,01	0,93 ± 0,00	0,82 ± 0,01	0,56 ± 0,00	0,63 ± 0,00
	1.5	0,92 ± 0,00	0,93 ± 0,00	0,80 ± 0,01	0,56 ± 0,00	0,63 ± 0,00
	2	0,92 ± 0,01	0,93 ± 0,00	0,79 ± 0,01	0,56 ± 0,00	0,63 ± 0,00
	2.5	0,92 ± 0,00	0,93 ± 0,00	0,78 ± 0,02	0,56 ± 0,00	0,63 ± 0,00
	3	0,91 ± 0,00	0,93 ± 0,00	0,78 ± 0,02	0,56 ± 0,00	0,63 ± 0,00
	5	0,91 ± 0,00	0,93 ± 0,00	0,78 ± 0,01	0,56 ± 0,00	0,63 ± 0,00
50	0	0,88 ± 0,00	0,94 ± 0,00	0,81 ± 0,00	0.58 ± 0,00	0,60 ± 0,00
	0.05	0,89 ± 0,00	0,96 ± 0,00	0,78 ± 0,00	0,55 ± 0,00	0,62 ± 0,00
	0.1	0,89 ± 0,00	0,95 ± 0,00	0,81 ± 0,00	0,55 ± 0,00	0,64 ± 0,00
	0.2	0,88 ± 0,00	0,95 ± 0,00	0,83 ± 0,00	0,56 ± 0,00	0,68 ± 0,01
	0.5	0,87 ± 0,01	0,95 ± 0,00	0,85 ± 0,00	0,56 ± 0,00	0,65 ± 0,00
	0.8	0,93 ± 0,01	0,95 ± 0,00	0,86 ± 0,00	0,56 ± 0,00	0,65 ± 0,00
	1	0,94 ± 0,01	0,95 ± 0,00	0,86 ± 0,00	0,55 ± 0,00	0,65 ± 0,00
	1.5	0,94 ± 0,01	0,95 ± 0,00	0,85 ± 0,01	0,55 ± 0,00	0,64 ± 0,01
	2	0,94 ± 0,01	0,94 ± 0,00	0,83 ± 0,01	0,55 ± 0,00	0,64 ± 0,00
	2.5	0,94 ± 0,00	0,94 ± 0,00	0,82 ± 0,01	0,55 ± 0,00	0,64 ± 0,00
	3	0,94 ± 0,00	0,94 ± 0,00	0,80 ± 0,01	0,55 ± 0,00	0,64 ± 0,00
	5	0,94 ± 0,00	0,94 ± 0,00	0,76 ± 0,02	0,55 ± 0,00	0,64 ± 0,00

Tableau 4.4 – Indice de Rand moyen et intervalle de confiance à 95% en fonction de ξ pour 20 et 50 contraintes choisies aléatoirement pour les jeux de données de l'UCI.

avec 20 et 50 contraintes, une nette amélioration de la qualité de la partition est visible lorsque ξ est compris entre 1.5 et 2. Cette variation du réglage de ξ selon le jeu de données utilisé peut s'expliquer de manière simple. Prenons par exemple $\xi = 1$, ce qui revient à donner le même poids aux deux termes de la fonction objectif. Dans ce cas, les points contraints influencent fortement leur voisinage. C'est ce qui est recherché pour Iris, puisque la distance Euclidienne utilisée pour construire la matrice de dissimilarité n'est pas adaptée au jeu de données, et que la solution désirée est éloignée de la solution de EVCLUS. Cependant pour le jeu de données Wine qui utilise une distance adaptée, l'influence des points contraints est trop forte : ceci a pour conséquence le déplacement des frontières, comme dans le cas d'une surexploitation d'une zone réduite de l'espace des données (cf. figure 4.13).

Il est aussi possible d'observer pour tous les jeux de données que plus le nombre de contraintes augmente, plus la valeur de ξ doit être augmentée afin d'obtenir de bon résultats. Un nombre de contraintes élevé influence en effet toujours de manière positive la partition finale (sous l'hypothèse que les contraintes utilisées ne sont pas incohérentes les unes vis à vis des autres). Pour le reste de nos expériences, nous

C	ξ	Iris	Wine	Glass	Ionosphere	LettersIJL
100	0	0,88 ± 0,00	0,94 ± 0,00	0,81 ± 0,00	0.58 ± 0,00	0,60 ± 0,00
	0.05	0,89 ± 0,00	0,96 ± 0,00	0,79 ± 0,00	0,54 ± 0,00	0,62 ± 0,00
	0.1	0,89 ± 0,00	0,96 ± 0,00	0,82 ± 0,00	0,54 ± 0,00	0,63 ± 0,00
	0.2	0,89 ± 0,00	0,96 ± 0,00	0,85 ± 0,00	0,56 ± 0,00	0,72 ± 0,01
	0.5	0,87 ± 0,01	0,96 ± 0,00	0,89 ± 0,00	0,59 ± 0,00	0,72 ± 0,01
	0.8	0,94 ± 0,01	0,96 ± 0,00	0,90 ± 0,00	0,59 ± 0,01	0,68 ± 0,01
	1	0,96 ± 0,00	0,96 ± 0,00	0,90 ± 0,00	0,59 ± 0,01	0,67 ± 0,01
	1.5	0,97 ± 0,00	0,96 ± 0,00	0,90 ± 0,00	0,57 ± 0,01	0,65 ± 0,01
	2	0,96 ± 0,00	0,96 ± 0,00	0,90 ± 0,00	0,56 ± 0,01	0,64 ± 0,01
	2.5	0,96 ± 0,00	0,96 ± 0,00	0,91 ± 0,00	0,56 ± 0,01	0,64 ± 0,01
	3	0,96 ± 0,00	0,96 ± 0,00	0,90 ± 0,01	0,56 ± 0,01	0,64 ± 0,01
	5	0,96 ± 0,00	0,96 ± 0,00	0,80 ± 0,02	0,56 ± 0,01	0,63 ± 0,01
200	0	0,88 ± 0,00	0,94 ± 0,00	0,81 ± 0,00	0.58 ± 0,00	0,60 ± 0,00
	0.05	0,89 ± 0,00	0,96 ± 0,00	0,78 ± 0,00	0,54 ± 0,00	0,62 ± 0,00
	0.1	0,90 ± 0,00	0,97 ± 0,00	0,82 ± 0,00	0,53 ± 0,00	0,63 ± 0,00
	0.2	0,90 ± 0,00	0,98 ± 0,00	0,86 ± 0,00	0,56 ± 0,00	0,67 ± 0,01
	0.5	0,88 ± 0,00	0,98 ± 0,00	0,91 ± 0,00	0,73 ± 0,01	0,88 ± 0,01
	0.8	0,96 ± 0,01	0,98 ± 0,00	0,94 ± 0,00	0,78 ± 0,01	0,82 ± 0,01
	1	0,98 ± 0,00	0,98 ± 0,00	0,95 ± 0,00	0,77 ± 0,01	0,76 ± 0,01
	1.5	0,99 ± 0,00	0,98 ± 0,00	0,96 ± 0,00	0,71 ± 0,02	0,68 ± 0,01
	2	0,99 ± 0,00	0,98 ± 0,00	0,96 ± 0,00	0,66 ± 0,02	0,67 ± 0,01
	2.5	0,99 ± 0,00	0,98 ± 0,00	0,96 ± 0,00	0,63 ± 0,01	0,65 ± 0,01
	3	0,99 ± 0,00	0,98 ± 0,00	0,96 ± 0,00	0,60 ± 0,01	0,63 ± 0,01
	5	0,99 ± 0,00	0,98 ± 0,00	0,96 ± 0,00	0,59 ± 0,01	0,62 ± 0,01

Tableau 4.5 – Indice de Rand moyen et intervalle de confiance à 95% en fonction de ξ pour 100 et 200 contraintes choisies aléatoirement pour les jeux de données de l'UCI.

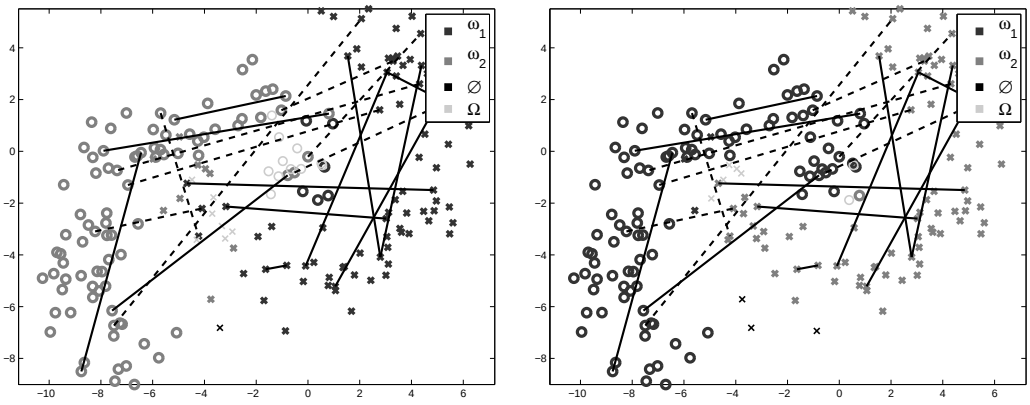
choisissons de prendre $\xi = 1$, quel que soit le nombre de contraintes.

4.4.2 Adaptation de la métrique

Les algorithmes EVCLUS et CEVCLUS sont susceptibles d'être peu performants dans le cas où les données relationnelles en entrée correspondent à une mesure de dissimilarité inadaptée à la structure sous-jacente du jeu de données. En admettant que seule la matrice de dissimilarités $\mathbf{D}$ est connue, il peut être intéressant de lui appliquer une transformation qui fasse ressortir cette structure. Pour cela, il est possible d'utiliser une fonction noyau. Prenons par exemple le jeu de données ToysBananas décrit au paragraphe 4.1.3. Comme le montre le tableau 4.6, l'indice de Rand obtenu avec EVCLUS est de 0.59. L'algorithme CEVCLUS avec 20 contraintes (cf. figure 4.16(a)) donne un indice de Rand de 0.77. Si un noyau gaussien avec une largeur de bande σ de 0.2 est utilisé, alors l'indice de Rand de EVCLUS est de 0.62 et l'indice de Rand de CEVCLUS (avec les mêmes contraintes que sans pré-traitement) est de 0.91 (cf. figure 4.6(b)). Notons que pour chaque expérience, ξ est fixé à 1.

Nb. cont.	aucun pré-traitement	noyau gaussien, $\sigma = 0.2$
0	0.59	0.62
20	0.77	0.91

Tableau 4.6 – Valeurs des indices de Rand obtenus par EVCLUS et CEVCLUS avec et sans pré-traitement de la matrice de dissimilarités pour le jeu de données ToysBananas.



(a) (b)

Figure 4.16 – Partitions crédales nettes de ToysBananas obtenues par CEVCLUS avec $\xi = 1$ et 20 contraintes sans utiliser de prétraitement (a) et en utilisant un noyau gaussien tel que $\sigma = 0.2$ (b). Les symboles représentent les classes réelles du jeu de données et les lignes continues (respectivement en pointillés) les contraintes Must-Link (respectivement Cannot-Link).

Nous pouvons illustrer l'avantage de cette stratégie sur un second exemple. Reprenons le jeu de données ToysDataVert généré automatiquement à l'aide de quatre Gaussiennes (cf. paragraphe 4.1.3). Un nouveau jeu de données, nommé ToysDataMoon, peut être créé en modifiant l'attribution précédente des classes : la première classe comprend désormais les données générées par une seule gaussienne et la seconde classe contient le reste des données (cf figure 4.17).

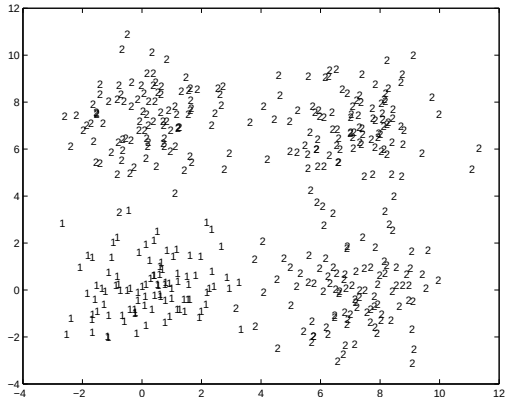
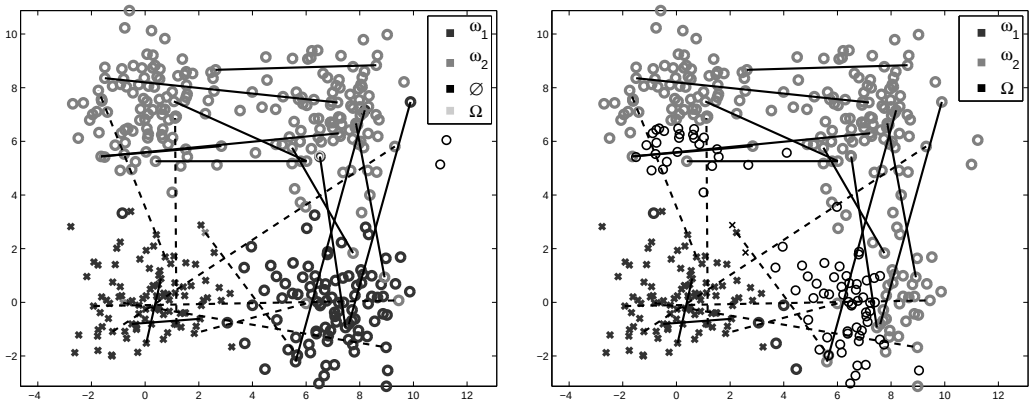


Figure 4.17 – Jeu de données ToysDataMoon généré automatiquement à partir de Gaussienne.

Sans pré-traitement, les algorithmes EVCLUS et CEVCLUS avec 20 contraintes ne permettent pas d'obtenir la partition escomptée (cf. table 4.7). En effet, les dissimilarités ne reflètent pas la structure des classes et un nombre limité de contraintes ne suffit pas à résoudre le problème. D'un point de vue numérique, le coût de conserver une séparation horizontale des données est plus faible que le coût de respecter les contraintes en modifiant la partition crédale du voisinage des objets contraints (cf. figure 4.18(a)). Un pré-traitement des données peut donc s'avérer intéressant. Nous utilisons un noyau polynomial, qui élève à la puissance d chaque élément de la matrice $\mathbf{D}$. Comme le montre la figure 4.18(b), fixer d à 2 et utiliser CEVCLUS avec 20 contraintes permet d'obtenir la solution désirée. L'indice de Rand obtenu est alors de 0.90. Il faut noter que le paramètre ξ est fixé à 1 et que chaque expérience est exécutée 10 fois, pour ensuite sélectionner la solution ayant la fonction objectif la plus basse.

Nb. cont.	aucun pré-traitement	noyau puissance, $d = 2$
0	0.63	0.63
20	0.62	0.90

Tableau 4.7 – Valeurs des indices de Rand obtenus par EVCLUS et CEVCLUS avec et sans modification préalable de la matrice de dissimilarités pour le jeu de données ToysDataMoon.



(a) (b)

Figure 4.18 – Partitions crédales nettes de ToysDataMoon obtenues par CEVCLUS avec $\xi = 1$ et 20 contraintes sans utiliser de prétraitement (a) et en utilisant un noyau puissance tel que $d = 2$ (b). Les symboles représentent les classes réelles du jeu de données et les lignes continues (respectivement en pointillés) les contraintes Must-Link (respectivement Cannot-Link).

4.4.3 Correction de la partition crédale

Il arrive souvent que la solution trouvée par CEVCLUS corresponde à un minimum local. C'est la raison pour laquelle l'algorithme doit être lancé plusieurs fois avec différentes initialisations. Selon les bases de données et les contraintes ajoutées, les minima locaux peuvent être plus ou moins nombreux. Nous avons ainsi pu constater que le problème est plus difficile dans le cas où les contraintes doivent guider l'algorithme vers une solution très différente de celle trouvée par EVCLUS.

C'est par exemple le cas de la base de données ToysDataVert pour laquelle EVCLUS trouve initialement une frontière verticale alors que la frontière doit être horizontale. Comme le montre la figure 4.19(a) qui est reprise des expériences de la partie 4.1.3, ajouter des contraintes à ToysDataVert permet de trouver une meilleure solution. Néanmoins cette solution comprend une contrainte Must-Link (verticale, à droite) dont les objets sont classées de manière inattendue : les voisins de ces points contraints sont en effet affectés à une classe différente. Ce problème étant dû à un minimum local, il est possible de le régler en lançant de nouveau l'algorithme CEVCLUS plusieurs fois. Cependant cette manière de procéder peut être coûteuse en temps, notamment si il existe un nombre important de contraintes.

Par conséquent, nous proposons un post-traitement permettant de corriger les erreurs de classification sur les contraintes les plus évidentes. Pour cela, le degré de conflit entre chaque point contraint et leurs voisins est étudié. Si les deux objets issus d'une même contrainte ont un degré de conflit élevé avec leurs voisins, alors les masses de croyance des différentes classes sont permutées afin de vérifier qu'il n'existe pas une meilleure solution que celle trouvée par CEVCLUS. L'algorithme 4.1 détaille ce procédé de correction.

Algorithme 4.1 : Pseudo-code de l'algorithme de post-traitement pour CEVCLUS.

Entrées : Distance $\mathbf{D}$, partition crédale $\mathbf{M}$, ensembles $\mathcal{M}$ et $\mathcal{C}$ de contraintes Must-Link et Cannot-Link, fonction objectif J, seuil de distance s_{dist}, seuil de conflit $s_{conflit}$.

Sorties : Partition crédale $\mathbf{M}$

début

  pour *chaque contrainte* $(\mathbf{o}_i, \mathbf{o}_j) \in \{\mathcal{M}, \mathcal{C}\}$ **faire**

    Trouver les voisins des objets $\mathbf{o}_i$ et $\mathbf{o}_j$:

$$v_i = \{\mathbf{o}_{i'} / d_{ii'} < s_{dist}\}, \quad v_j = \{\mathbf{o}_{j'} / d_{jj'} < s_{dist}\}.$$

    Calculer le conflit des objets $\mathbf{o}_i$ et $\mathbf{o}_j$ avec leurs voisins respectif :

$$K_i = \frac{1}{n_i} \sum_{o_{i'} \in v_i} K_{ii'}, \quad K_j = \frac{1}{n_j} \sum_{o_{j'} \in v_j} K_{jj'},$$

    avec n_i, n_j le nombre d'objets pour les sous-ensembles v_i, v_j.

    si $K_i > s_{conflit}$ *et* $K_j > s_{conflit}$ **alors**

      pour *chaque classe* $c_1, c_2 = 1 \dots c,\ c_2 > c_1$ **faire**

        Permuter les classes c_1 et c_2 pour obtenir une nouvelle matrice $\mathbf{M}_{new}$

        Calculer la nouvelle fonction objectif J_{new}

        si $J_{new} < J$ **alors**

          $\mathbf{M} \leftarrow \mathbf{M}_{new}$

          $J \leftarrow J_{new}$

fin

La figure 4.19(b) montre la partition crédale nette obtenue en appliquant un post-traitement à la solution obtenue par CEVCLUS avec 10 contraintes (cf. fi-

gure 4.19(a)). Il est possible d'observer que les erreurs sur les points contraints ont été corrigées.

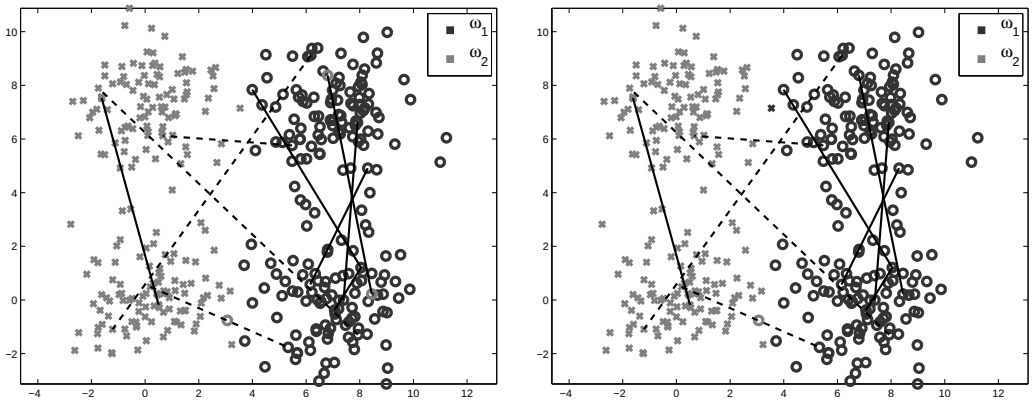
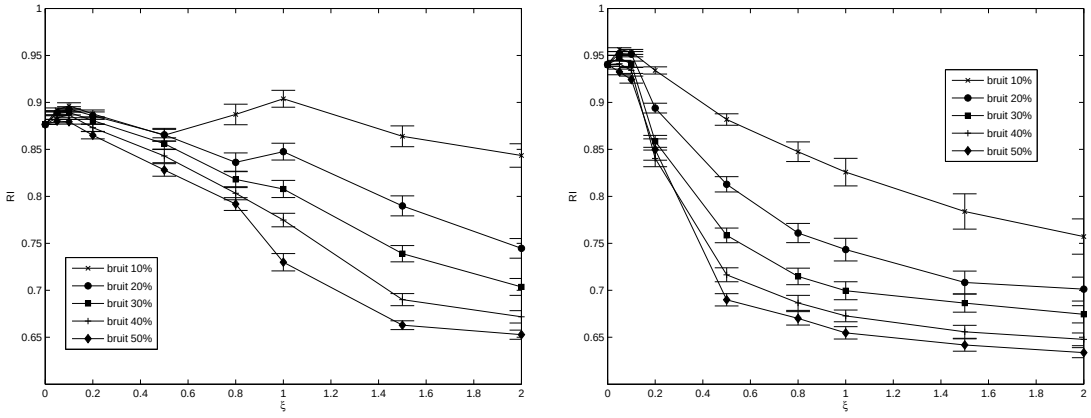


Figure 4.19 – Partitions crédales nettes obtenues pour ToysDataVert en utilisant CEVCLUS avec 10 contraintes (a) et CEVCLUS avec 10 contraintes suivi d'un post-traitement (b). Les symboles représentent les classes réelles et les lignes continues et en pointillés représentent les contraintes Must-Link et Cannot-Link.

4.4.4 Robustesse aux contraintes bruitées

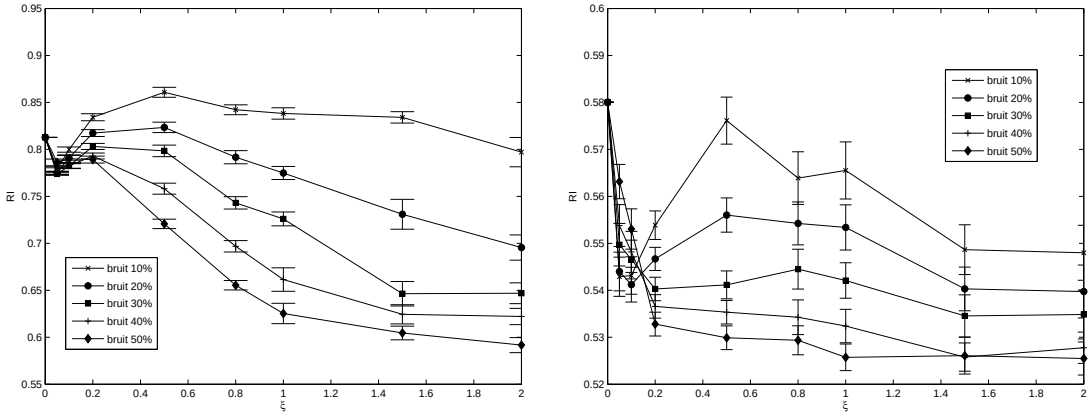
Il peut arriver que les contraintes soient incohérentes entre elles. Par exemple, la spécification de règles permettant de créer automatiquement des contraintes peut ne pas être totalement satisfaisante. Nous avons vu en effet pour le jeu de données 20Newsgroups que ces règles produisent 4.1% de contraintes dont le type (Must-Link ou Cannot-Link) est incorrectement choisi. De même, il peut arriver en apprentissage actif que l'expert se trompe sur le lien réel existant entre deux objets. Il est donc important de connaître le comportement de l'algorithme CEVCLUS dans le cas de contraintes bruitées. Pour cela, nous utilisons le même protocole expérimental que pour l'algorithme CECM : après avoir fixé le nombre de contraintes à 100, nous introduisons du bruit aléatoirement. Ainsi, une contrainte Must-Link (respectivement Cannot-Link) est changé en contrainte Cannot-Link (respectivement Must-Link) avec une probabilité comprise entre 0.1 et 0.5. Les figures 4.20, 4.21 et 4.22 présente la moyenne calculée sur 100 expériences pour chaque valeur de ξ pris en compte et chaque pourcentage de bruit.

Il est possible de remarquer que la robustesse de l'algorithme par rapport au bruit dépend essentiellement du jeu de données. Les jeux de données Glass et LettersIJL obtiennent en effet une amélioration des résultats dans le cas d'un pourcentage faible de bruit. Les autres jeux de données (Iris, Wine et Ionosphere) sont par contre plus sensibles au bruit. Pour Iris et Wine, seule une très faible valeur de ξ permet d'améliorer légèrement les résultats. Il est donc essentiel de détecter les contradictions dans les contraintes, afin de réduire la valeur utilisée de ξ ou de supprimer toutes les contraintes incohérentes.



(a) (b)

Figure 4.20 – Indice de Rand moyen en fonction de ξ pour 100 contraintes choisies aléatoirement et bruitées entre 10% et 50%, pour les jeux de données Iris (a) et Wine (b).



(a) (b)

Figure 4.21 – Indice de Rand moyen en fonction de ξ pour 100 contraintes choisies aléatoirement et bruitées entre 10% et 50%, pour les jeux de données Glass (a) et Ionosphere (b).

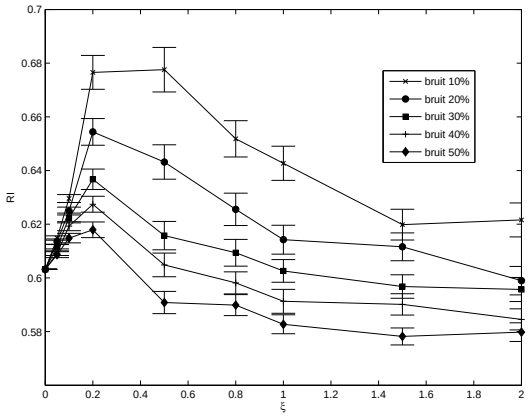


Figure 4.22 – Indice de Rand moyen en fonction de ξ pour 100 contraintes choisies aléatoirement et bruitées entre 10% et 50%, pour le jeu de données LettersIJL.

4.4.5 Comparaison des méthodes

L'algorithme CEVCLUS est une méthode de classification par contraintes qui utilise des données relationnelles. Il peut être intéressant de le comparer avec un algorithme similaire. Les figures 4.23, 4.24 et 4.25 présentent donc les indices de Rand des algorithmes CEVCLUS et CCL (cf. paragraphe 1.2.1) en fonction d'un nombre de contraintes variant de 0 à 200. Notons que les contraintes sont choisies aléatoirement et que le paramètre ξ de CEVCLUS est fixé à 1.

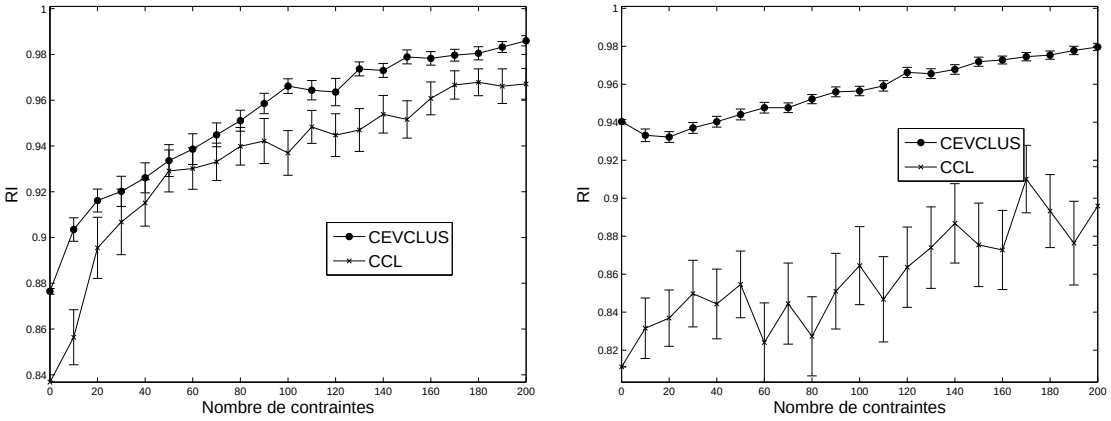


Figure 4.23 – Comparaison des indices de Rand moyens et intervalles de confiance à 95% sur 100 essais pour CEVCLUS et CCL pour les jeux de données Iris (a) et Wine (b).

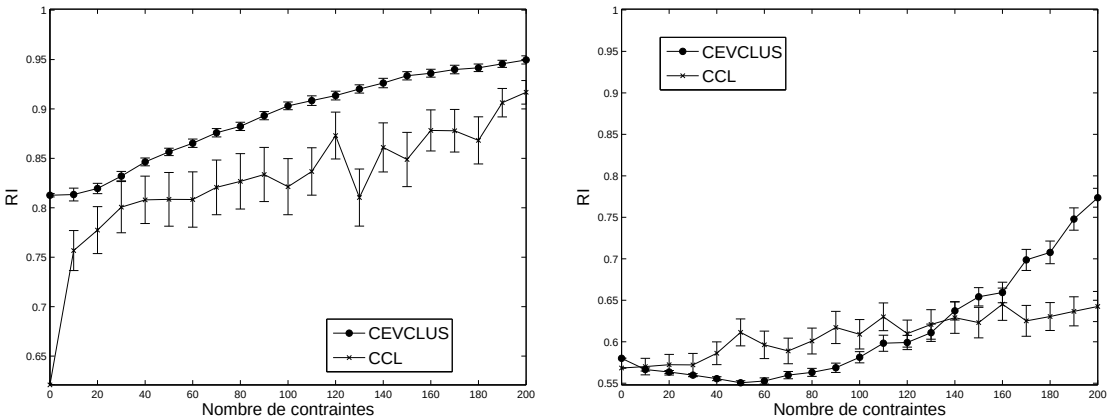


Figure 4.24 – Comparaison des indices de Rand moyens et intervalles de confiance à 95% sur 100 essais pour CEVCLUS et CCL pour les jeux de données Glass (a) et Ionosphere (b).

Les figures 4.23, 4.24 et 4.25 montrent que l'algorithme CEVCLUS donne en général de meilleurs résultats. En effet, CCL est plus performant sur le jeu de données Ionosphere avec un nombre de contraintes inférieur à 150 seulement. Il est possible d'observer par ailleurs que les intervalles de confiance sont relativement élevés pour l'algorithme CCL. Contrairement à CEVCLUS, l'algorithme CCL est en effet très

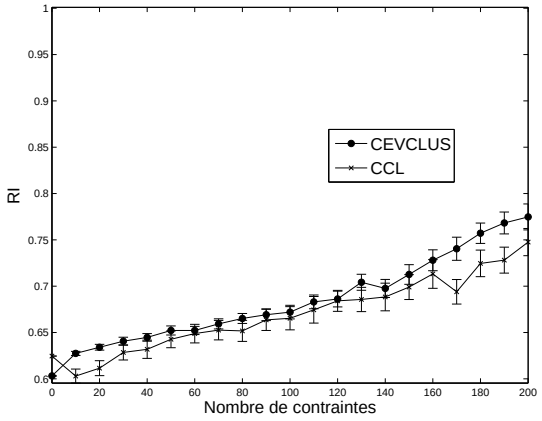


Figure 4.25 – Comparaison des indices de Rand moyens et intervalles de confiance à 95% sur 100 essais pour CEVCLUS et CCL pour le jeu de données LettersIJL.

sensible au jeu de contraintes pris en compte. Il faut noter malgré tout que le temps d'exécution de CCL est moins important que le temps d'exécution de CEVCLUS.

4.4.6 Complexité

Tout comme CECM, la complexité de l'algorithme CEVCLUS est linéaire par rapport au nombre d'individus à classer et exponentiel par rapport au nombre de classes. La méthode consistant en une descente de gradient, CEVCLUS reste limité à un faible nombre de classes et d'individus (5 classes et 1000 objets semblent être un seuil à ne pas dépasser).

Tout comme CECM, la complexité de CEVCLUS peut être réduite grâce à la suppression d'une partie des sous-ensembles $A_j \subseteq \Omega$. Par exemple, si seuls les singletons, l'ensemble vide et Ω sont conservés, alors la complexité de l'algorithme devient linéaire par rapport au nombre de classes. De cette manière, il existe un compromis entre le temps d'exécution de CEVCLUS et la richesse des informations contenue dans la partition crédale finale. Pour illustrer ce principe, le jeu de données ToysDataVert est repris et le nombre de classes est fixé à 4 afin que chaque Gaussienne corresponde à une classe. Deux versions de l'algorithme CEVCLUS avec $\xi = 1$ et 100 contraintes sont alors exécutées. Elles prennent toutes les deux la partition de EVCLUS comme partition crédale initiale. La première version, nommée CEVCLUS-1, utilise tous les sous-ensembles de Ω disponibles alors que la seconde version, nommée CEVCLUS-2, correspond à une version limitée où seuls les singletons, l'ensemble vide et Ω sont sélectionnés. Le tableau 4.8 montre les temps de calculs de ces expériences moyennés sur 20 essais. Les intervalles de confiance à 95% sont aussi présentés. Il est possible d'observer que CEVCLUS-2 est plus rapide que CEVCLUS-1. Sachant que l'indice de Rand moyen obtenue est de 0.99 pour les deux versions de CEVCLUS, nous pouvons déduire que la limitation du nombre de sous-ensembles permet dans certain cas d'augmenter la rapidité de l'algorithme sans pour autant diminuer ses performances. Toutefois il faut noter que les temps de calcul restent élevés. En comparaison, l'algorithme CECM est en effet nettement plus rapide (cf. tableau 3.12).

	CEVCLUS-1	CEVCLUS-2
CPU(s)	3752.20 ± 482.22	2756.13 ± 94.73
Nb. Ité.	503.65 ± 64.54	483.65 ± 16.78

Tableau 4.8 – Comparaison entre la version complète (CEVCLUS-1) et la version limitée (CEVCLUS-2) pour l'algorithme CEVCLUS appliqué sur le jeu de données ToysDataVert avec $c = 4$. La moyenne et l'écart type pour le temps CPU et le nombre d'itérations sont calculés sur 20 essais pour 100 contraintes choisies aléatoirement.

Synthèse du chapitre

Dans ce chapitre, nous avons présenté une nouvelle méthode de classification basée sur l'algorithme évidentiel EVCLUS. Ce nouvel algorithme, nommé Constrained-EVCLUS (ou plus simplement CEVCLUS), traite des données relationnelles et intègre sous forme de contraintes des connaissances a priori afin d'améliorer les résultats de classification.

Un série d'expériences menée sur des jeux de données de l'UCI prouvent que l'ajout de contraintes permet d'améliorer la qualité de la partition, notamment lorsque la matrice de dissimilarité correspond à une métrique inadaptée au problème. Une application de l'algorithme CEVCLUS a ensuite été proposée avec le jeu de données 20Newsgroups. Elle a permis de mettre en avant la possibilité d'obtenir automatiquement un certain nombre de contraintes grâce aux connaissances d'un expert. Dans le cas où il est impossible d'obtenir ces contraintes de manière automatique, une procédure d'apprentissage actif peut être mise en place. Cette dernière consulte un utilisateur pour connaître le type de lien (Must-Link ou Cannot-Link) d'un ensemble restreint de contraintes choisi automatiquement. L'apprentissage actif a été testé dans le cadre d'une application de segmentation d'images.

Afin d'améliorer les résultats de CEVCLUS, deux procédures ont ensuite été proposées. La première, intégrée en amont de l'algorithme de classification, transforme la matrice de dissimilarité à l'aide d'une fonction noyau. Cette transformation doit permettre de faire ressortir plus aisément la structure sous-jacente des données. La seconde méthode consiste à corriger la partition crédale trouvée par CEVCLUS. L'algorithme convergeant en effet parfois vers un minimum local, il est intéressant d'étudier la cohérence de la partition crédale trouvée.

Enfin, les performances de CEVCLUS ont été comparées avec l'algorithme de classification par contraintes CCL. Les résultats montrent que CEVCLUS est meilleur pour la plupart des jeux de données. De plus, l'ensemble des contraintes sélectionnées influence moins les résultats de CEVCLUS que ceux de CCL. Néanmoins, il faut noter que la complexité de l'algorithme CEVCLUS est élevée. Pour obtenir une classification dans des temps raisonnables, les jeux de données ne doivent donc pas comprendre un nombre important de classes et d'individus. Cette limitation peut cependant être modérée par la réduction du nombre de sous-ensemble utilisé dans l'algorithme CEVCLUS.

Conclusions et perspectives

Conclusions

Les travaux réalisés dans ce mémoire se situent à l'intersection de deux grandes familles de classification automatique. La première, nommée classification sous contraintes, désigne les algorithmes qui incorporent des connaissances a priori sous forme de contraintes dans le processus de classification. Ces connaissances, aisément extraites du domaine de l'expert, permettent d'améliorer la qualité de la partition finale. La seconde famille, nommée classification évidentielle, correspond aux méthodes qui utilisent le cadre des fonctions de croyance afin de décrire la structure des données. La partition crédale, riche en informations, permet d'exprimer de manière naturelle le doute sur l'affectation des objets aux classes. L'expert peut alors prendre une décision en tenant compte de ces incertitudes. Afin de tirer parti des avantages de chacune des deux familles de classification, nous avons proposé dans ce mémoire de les combiner. Ainsi, nous recherchons un résultat de classification à la fois riche en informations et fiable.

Dans ce cadre, deux nouveaux algorithmes de classification sous contraintes ont été proposés. Le premier, nommé CECM, est une extension de l'algorithme évidentiel ECM. Il comprend deux modifications majeures par rapport à sa version originale : l'ajout de contraintes par paires et l'introduction d'une métrique adaptative. La nouvelle distance, plus générale que la distance Euclidienne utilisée dans ECM, permet de constituer des classes non sphériques. L'ajout de contraintes permet en autre d'ajuster la métrique. Le deuxième algorithme, nommé CEVCLUS, correspond à une variante de la méthode EVCLUS permettant l'incorporation de connaissance a priori sous forme de contraintes définies sur des paires d'objets.

Dans les deux cas, l'ajout de contraintes permet aux nouveaux algorithmes d'obtenir de meilleurs résultats que leurs algorithmes de base. En effet, introduire des connaissances a priori permet de guider les algorithmes CECM et CEVCLUS vers la solution désirée. Les nouveaux algorithmes ont été comparés à diverses méthodes de classification sous contraintes. La qualité des partitions obtenues par CECM et CEVCLUS est souvent meilleure que pour les autres algorithmes, ce qui prouve que la richesse d'informations apportée par le modèle évidentiel contribue à l'obtention de bons résultats.

Cependant, calculer une partition crédale peut s'avérer coûteux en temps de calcul. En effet, cela implique une augmentation exponentielle de la complexité par rapport au nombre de classes. Par conséquent, soit ce nombre de classes doit être limité, soit

le nombre de sous-ensembles construit à partir des classes doit être réduit. Ainsi, il existe un compromis entre une solution de bonne qualité et son temps d'exécution.

Dans le cas où il est possible de créer des contraintes Must-Link et Cannot-Link à partir de connaissances a priori, un expert pourra employer les algorithmes proposés dans ce mémoire. L'un sera plus aisé à utiliser ou donnera de meilleurs résultats que l'autre selon les situations. Par exemple, CEVCLUS n'a besoin que d'une matrice de dissimilarité pour être exécuté. Il peut donc être utilisé par un plus grand nombre de jeux de données que l'algorithme ECM qui nécessite des données vectorielles. De plus, il nécessite uniquement de régler de l'hyperparamètre correspondant au compromis entre la structure des données et les contraintes. Le fait d'avoir peu d'hyperparamètres à régler permet à un utilisateur de ne pas avoir à chercher comment les ajuster au mieux. Enfin, CEVCLUS s'avère mieux adapté que CECM à la reconnaissance de classes non linéairement séparables. D'un autre côté, CECM est parfaitement adapté aux données ayant des classes de formes ellipsoïdales. Il est aussi plus rapide et moins sujet aux problèmes de minima locaux que CEVCLUS.

Selon les connaissances a priori utilisées pour un algorithme de classification sous contraintes, la qualité de la partition peut varier de manière importante, notamment pour un nombre de contraintes faible. Il existe donc des contraintes qui ont un impact variable sur la partition finale. Ainsi, la classification sous contraintes a donné naissance à une famille d'algorithmes nommée apprentissage actif. Ces algorithmes permettent de sélectionner automatiquement des contraintes permettant d'obtenir le meilleur résultat de classification possible. Nous avons proposé pour les méthodes ECM et EVCLUS deux stratégies d'apprentissage actif. La première, utilisée pour CECM, permet de démontrer l'intérêt d'exploiter certaines contraintes plutôt que d'autres. La seconde stratégie, utilisée avec CEVCLUS, a été créée afin d'être opérationnelle pour une application réelle. Elle permet à un utilisateur de choisir les contraintes qui lui semblent les plus appropriées.

Perspectives

Les travaux présentés dans ce mémoire ouvrent de nombreuses pistes de recherche, aussi bien à court terme qu'à long terme.

Tout d'abord à court terme, l'étude de l'algorithme CEVCLUS pourrait être approfondie. Une étude et une comparaison de l'apprentissage actif de CECM et CEVCLUS pourrait être développée. En effet, les deux algorithmes procèdent aux regroupements de données de manière différente. Il semble alors évident que la stratégie d'apprentissage actif doit elle aussi être différente. Enfin, la réalisation d'expériences pour l'algorithme de classification relationnel SS-CARD permettrait de renforcer la partie comparaison de CEVCLUS de ce mémoire.

À long terme, plusieurs travaux peuvent être proposés. Tout d'abord, les contraintes ajoutées dans les algorithmes proposés sont dures, c'est-à-dire qu'elles sont considérées comme totalement fiables. L'hyperparamètre gérant le compromis entre

les contraintes et la structure sous-jacente des données permet ensuite de prendre en compte les contraintes de manière douce. Néanmoins, un expert peut parfois quantifier le degré d'incertitude d'une contrainte. Cette information pourrait être intégrée à l'algorithme sous la forme d'un degré de plausibilité associé à la contrainte. Une seconde piste de travail peut être considérée : l'introduction de nouvelles formes de contraintes dans les algorithmes de classification évidentielle ECM et EVCLUS. Par exemple, des contraintes spatiales semblent être une forme intéressante de contraintes dans le cadre de segmentation d'images.

Enfin, dans le cadre de l'apprentissage actif, de nombreux travaux peuvent être menés. Ce concept a en effet été appliqué en classification semi-supervisée mais n'a pratiquement pas été étudié en classification sous contraintes. Ainsi, il serait intéressant de suggérer de nouvelles stratégies d'apprentissage actif en se basant sur celles proposées en classification semi-supervisée. Par exemple, pour éviter une chute de performance due à la surexploitation de l'espace d'une région particulière des données, il est possible de demander à l'apprentissage actif de rechercher des contraintes dans d'autres régions. Il faut noter par ailleurs que le cadre théorique des fonctions de croyance rend les algorithmes CECM et CEVCLUS particulièrement adaptés à l'utilisation de l'apprentissage actif. Cette stratégie nécessite a en effet de recueillir des informations sur les données à classifier afin de proposer des paires de contraintes. La partition crédale issue de la classification évidentielle semble alors appropriée : riche en informations, elle permet de mieux cerner les objets incertains ou imprécis que d'autres types de partitions telles que la partition dure ou la partition floue. Finalement, d'autres pistes de recherche dérivées de l'apprentissage peuvent être intéressantes à étudier. En effet, si par exemple plusieurs contraintes apportent la même information, alors il est possible d'en supprimer quelques unes. La rapidité de l'algorithme peut ainsi être augmentée.

Bibliographie

[1] Y. Abu-Mostafa. Machines that learn from hints. *Scientific American*, 272(4) : 64–69, 1995.

[2] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6 (1) :937–965, June 2005.

[3] S. Basu, A. Banerjee, and R. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 333–344, Lake Buena Vista, FL, USA, April 2004.

[4] S. Basu, I. Davidson, and K. Wagstaff. *Constrained clustering : Advances in algorithms, theory, and applications*. Chapman & Hall/CRC, 2008.

[5] J. Bezdek and R. Hathaway. Vat : A tool for visual assessment of (cluster) tendency. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN'02)*, volume 3, pages 2225–2230, Hawaii, USA, May 2002. IEEE.

[6] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981. ISBN 0306406713.

[7] M. Bilenko, S. Basu, and R. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning*, volume 69, Banff, Alberta, Canada, July 2004.

[8] I. Bloch. Defining belief functions using mathematical morphology : Application to image fusion under imprecision. *International Journal of Approximate Reasoning*, 48 :437–465, 2008.

[9] I. Borg and P. Groenen. *Modern Multidimensional Scaling*. Springer (New York), December, 1997.

[10] H. Bunke and U. Bühler. Applications of approximate string matching to 2d shape recognition. *Pattern recognition*, 26(12) :1797–1812, 1993.

[11] C. Cardie and K. Wagstaff. Noun phrase coreference as clustering. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 82–89, College Park, MD, USA, June 1999.

[12] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms, Second Edition*, chapter Graph algorithm, pages 525–643. MIT Press and McGraw-Hill, 2001.

[13] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 208–215, Helsinki, Finland, July 2008. ACM.

[14] R. Davé. Clustering relational data containing noise and outliers. *Pattern Recognition Letters*, 12 :657–664, 1991.

[15] I. Davidson and S. Ravi. Clustering with constraints : Feasibility issues and the k-means algorithm. In *Proceedings of the 5th SIAM International Conference on Data Mining*, pages 138–149, Newport Beach, CA, USA, April 2005. Society for Industrial Mathematics.

[16] I. Davidson, K. Wagstaff, and S. Basu. Measuring constraint-set utility for partitional clustering algorithms. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, volume 4213, pages 115–126, Berlin, Germany, September 2006.

[17] A. Dempster. Upper and lower probabilities induced by multivalued mapping. *Annals of Mathematical Statistics*, AMS-38 :325–339, 1967.

[18] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1) :1–38, 1977.

[19] T. Denœux. A k-nearest neighbor classification rule based on Dempster-Shafer theory. *IEEE Transactions on Systems, Man and Cybernetics*, 25(5) :804–813, 1995.

[20] T. Denœux. A neural network classifier based on Dempster-Shafer theory. *IEEE Transactions on Systems, Man and Cybernetics : A*, 30(2) :131–150, 2000.

[21] T. Denœux and M. Masson. EVCLUS : evidential clustering of proximity data. *IEEE Transactions on Systems, Man and Cybernetics : B*, 34(1) :95–109, 2004.

[22] D. Dubois, H. Prade, and H. Farreny. *Théorie des possibilités : applications à la représentation des connaissances en informatique*. Masson Paris, 1988. ISBN 222581273X.

[23] J. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Cybernetics and Systems*, 3(3) :32–57, 1973.

[24] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*, chapter Hierarchical clustering, pages 55–89. Wiley, 4th edition, january 2009.

[25] H. Frigui and C. Hwang. Adaptive concept learning through clustering and aggregation of relational data. In *Proceedings of the 7th SIAM International Conference on Data Mining*, 2007.

[26] H. Frigui and R. Krishnapuram. Clustering by competitive agglomeration. *Pattern Recognition*, 30(7) :1109–1119, 1997.

[27] C. Gasperin. Active learning for anaphora resolution. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 1–8, Boulder, Colorado, USA, June 2009. Association for Computational Linguistics.

[28] I. Gath and A. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7) :773–780, 1989.

[29] A. Gattein and P. Vannoorenberghe. A comparative analysis of two approches using the road network for tracking ground targets. In *Proceedings of the 7th International Conference of Information Fusion*, Stockholm, Sweden, June 2004.

[30] D. Gondek and T. Hofmann. Non-redundant data clustering. *Knowledge and Information Systems*, 12(1) :1–24, 2007.

[31] N. Grira, M. Crucianu, and N. Boujemaa. Active semi-supervised fuzzy clustering. *Pattern Recognition*, 41(5) :1834–1844, 2008.

[32] Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS)*, number 20, pages 593–600, Vancouver, British Columbia, Canada, December 2007. MIT Press.

[33] D. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proceeding of the IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, volume 17, pages 761–765, 1978.

[34] R. Hartley. Transmission of information. *The Bell System Technical Journal*, 7(3) :535–563, 1928.

[35] R. Hathaway and J. Bezdek. Nerf c-means : Non-euclidiean relational fuzzy clustering. *Pattern Recognition*, 27(3) :429–437, 1994.

[36] R. Hathaway, J. Davenport, and J. Bezdek. Relational duals of the c-means clustering algorithms. *Pattern Recognition*, 22(2) :205 – 212, 1989.

[37] S. Hoi, R. Jin, and M. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *Proceedings of the 24th International Conference on Machine Learning*, pages 361–368. ACM, 2007.

[38] D. Klein, S. Kamvar, and C. Manning. From instance-level constraints to space-level constraints : Making the most of prior knowledge in data clustering. In *Proceedings of the 19th International Conference on Machine Learning*, pages 307–314, Sydney, Australia, July 2002.

[39] G. Klir and M. Wierman. *Uncertainty-based information : Elements of generalized information theory*. Springer Verlag, New York, 1999.

[40] T. Kohonen. *Self-organizing Maps*. Springer, Berlin, 1997.

[41] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering : a kernel approach. *Machine Learning*, 74(1) :1–22, 2009.

[42] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2) :129–137, 1982.

[43] L. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, December 1967.

[44] M.-H. Masson and T. Denœux. ECM : An evidential version of the fuzzy c-means algorithm. *Pattern Recognition*, 41(4) :1384–1397, 2008.

[45] M.-H. Masson and T. Denœux. RECM : Relational evidential c-means algorithm. *Pattern Recognition Letters*, 30(11) :1015–1026, 2009. ISSN 0167-8655.

[46] J. Mauclair and J. Pinquier. Fusion of descriptors for speech / music classification. In *Proceedings of the 12th European Conference on Signal Processing (EUSIPCO)*, pages 1285–1288, Vienna, Austria, September 2004.

[47] E. Pekalska and R. Duin. *The Dissimilarity Representation for Pattern Recognition*, volume 64. Singapore, foundations and applications, world scientific edition, 2005.

[48] D. Pelleg and D. Baras. K-means with large and noisy constraint sets. In *Proceedings of the 18th European Conference on Machine Learning*, pages 674–682, Warsaw, Poland, September 2007. Springer.

[49] M. Rombaut and Y. Zhu. Study of dempster-shafer theory for image segmentation applications. *Image and Vision Computing*, 20 :15–23, 2002.

[50] R. S., W. J., and T.-S. C. A novel approach to auto image annotation based on pairwise constrained clustering and semi-naïve bayesian model. In *Proceedings of the 11th International Multimedia Modelling Conference (MMM'05)*, pages 322–327, 2005.

[51] J. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18 :401–409, 1969. ISSN 0018-9340.

[52] A. Schein and L. Ungar. Active learning for logistic regression : an evaluation. *Machine Learning*, 68(3) :235–265, 2007.

[53] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Proceedings of the 16th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, Vancouver and Whistler, British Columbia, Canada, December 2003. Bradford Book.

[54] G. Shafer. *A mathematical theory of evidence*. Princeton university press, Princeton, NJ, 1976.

[55] F. Silva and L. Almeida. Speeding up backpropagation. *Advanced neural computers*, 3 :151–158, 1990.

[56] P. Smets. Combination of non distinct evidences. In *Proceedings of the 1984 American Control Conference*, pages 554–555, 1984.

[57] P. Smets. The combination of evidence in the transferable belief model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5) :447–458, 1990.

[58] P. Smets. Constructing the pignistic probability function in a context of uncertainty. In M. Henrion, R. D. Shachter, L. N. Kanal, and J. F. Lemmer, editors, *Proceeding in the 5th Uncertainty in Artificial Intelligence*, pages 29–40. North Holland, Amsterdam, 1990.

[59] P. Smets. The application of the transferable belief model to diagnostic problems. *International Journal of Intelligent Systems*, 13 :127–157, 1998.

[60] P. Smets. Data fusion in the transferable belief model. In *Proceedings of the 3rd International Conference of Information Fusion*, pages 21–33, Paris, France, July 2000.

[61] P. Smets and R. Kennes. The transferable belief model. *Artificial Intelligence*, 66 :191–234, 1994.

[62] B. Solaiman, L. Lecornu, and C. Roux. Edge detection through information fusion using fuzzy and evidential reasoning concepts. In *Proceedings in Sensor Fusion : Architectures, Algorithms and Application IV*, volume 4051, pages 267–278, Orlando, USA, April 2000.

[63] K. Wagstaff. Value, cost, and sharing : Open issues in constrained clustering. In *Knowledge Discovery in Inductive Databases (KDID)*, volume 4747, pages 1–10, Berlin, Germany, September 2006. Springer.

[64] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning*, pages 577–584, Williamstown, MA, USA, July 2001.

[65] P. Walley. *Statistical reasoning with imprecise probabilities*. Chapman and Hall, London, 1991.

[66] M. Windham. Numerical classification of proximity data with assignment measures. *Journal of classification*, 2(1) :157–172, 1985.

[67] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Proceedings of the 15th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 521–528, Vancouver, British Columbia, Canada, December 2002.

[68] R. Yager. On the normalization of fuzzy belief structures. *International Journal of Approximate Reasoning*, 14(2-3) :127–153, 1996.

[69] B. Yan and C. Domeniconi. Kernel optimization using pairwise constraints for semi-supervised clustering. Technical Report ISE-TR-06-09, George Mason University, 2006.

[70] R. Yan, J. Zhang, and A. Hauptmann. A discriminative learning framework with pairwise constraints for video object classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 284–291, 2004.

[71] Y. Ye and E. Tse. An extension of Karmarkar's projective algorithm for convex quadratic programming. *Mathematical Programming*, 44(1) :157–179, 1989.

[72] L. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1 :3–28, 1978.

[73] E. Zeng, Y. Chengyong, L. Tao, and G. Narasimhan. On the effectiveness of constraints sets in clustering genes. In *IEEE International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 79–86, 2007.

[74] S. Zhong and J. Ghosh. Scalable, balanced model-based clustering. In *Proceedings of the 3rd SIAM International Conference on Data Mining*, pages 71–82, San Francisco, CA, usa, May 2003.

Première partie

Annexes

Algorithmes de classification automatique semi-supervisée

Le tableau A.1 présente les différents algorithmes de classification automatique semi-supervisée basés sur les c-moyennes. La colonne prétraitement sépare les algorithmes utilisant la connaissance a priori en amont de la classification des autres algorithmes. Les croix indiquent la possibilité pour les algorithmes de satisfaire ou non les propriétés décrites par les colonnes. Ainsi COP peut effectuer un prétraitement avant la classification, PC n'est pas forcé de respecter totalement les contraintes Must-Link et Cannot-Link, et DML peut utiliser n'importe quel algorithme de classification automatique une fois le prétraitement réalisé.

Les deux algorithmes CECM et CEVCLUS, séparés par une double ligne, correspondent aux nouvelles méthodes proposées dans le cadre de la classification automatique semi-supervisée (cf. chapitres 3 et 4).

algorithme	prétraitement	base algorithmique	données	respect total des contraintes	modification des distances
COP	×	c-moyenne	vectorielle	Oui	Non
PC	×	c-moyenne	vectorielle	×	Non
MPC	×	c-moyenne	vectorielle	×	Oui
PCCA	×	FCM + CA	vectorielle	×	Oui
SSCARD	×	RFCM	relationnelle	×	Non
CCL	Oui	CL	relationnelle	Oui	Oui
DML	Oui	×	vectorielle	×	Oui
KG	Oui	×	relationnelle	×	Oui
CECM	×	ECM	vectorielle	×	Oui
CEVCLUS	×	EVCLUS	relationnelle	×	Non

Tableau A.1 – Algorithmes de classification automatique semi-supervisée basés sur les c-moyennes et utilisant des contraintes Must-Link / Cannot-Link.

Équations de mise à jour pour l'algorithme ECM

Pour résoudre le problème de minimisation de la fonction objectif J_{ECM} (cf. équation (2.16)) par rapport à la partition crédale $\mathbf{M}$, n multiplicateurs de Lagrange λ_i sont introduits :

$$\mathcal{L}(\mathbf{M}, \lambda_1, \ldots, \lambda_n) = J_{ECM}(\mathbf{M}, \mathbf{V}) - \sum_{i=1}^{n} \lambda_i \left(\sum_{j/A_j \subseteq \Omega, A_j \neq \emptyset} m_{ij} + m_{i\emptyset} - 1 \right). \quad \text{(B.1)}$$

En considérant les centres de gravité $\mathbf{V}$ à fixes, l'annulation des dérivées partielles de Lagrange par rapport à m_{ij}, $m_{i\emptyset}$, λ_i permet d'obtenir les équations de mise à jour des masses suivantes :

$$m_{ij} = \frac{|A_j|^{-\alpha/(\beta-1)} d_{ij}^{-2/(\beta-1)}}{\sum_{A_k \neq \emptyset} |A_k|^{-\alpha/(\beta-1)} d_{ik}^{-2/(\beta-1)} + \rho^{-2/(\beta-1)}} \qquad \forall i = \{1, \ldots, n\},\ A_j \neq \emptyset, \quad \text{(B.2)}$$

et

$$m_{i\emptyset} = 1 - \sum_{A_j \neq \emptyset} m_{ij} \qquad \forall i = \{1, \ldots, n\}. \quad \text{(B.3)}$$

Dans un second temps, $\mathbf{M}$ est fixé afin de trouver l'équation de mise à jour des centres de gravité. La minimisation de J_{ECM} par rapport à $\mathbf{V}$ est un problème non contraint. Les conditions d'optimalité sont donc trouvées en annulant les dérivées partielles de la fonction objectif par rapport à $\mathbf{V}$. Cela amène à résoudre un système d'équations linéaires à chaque étape de l'agorithme ECM. Soient $\mathbf{B}$ une matrice de taille $(c \times p)$ définie telle que :

$$\mathbf{B}_{lq} = \sum_{i=1}^{n} x_{iq} \sum_{A_j \neq \emptyset} |A_j|^{\alpha-1} m_{ij}^{\beta} s_{lj} = \sum_{i=1}^{n} x_{iq} \sum_{A_j \ni \omega_l} |A_j|^{\alpha-1} m_{ij}^{\beta} \qquad \forall l = \{1, \ldots, c\},\ q = \{1, \ldots, p\} \quad \text{(B.4)}$$

et $\mathbf{H}$ une matrice de taille $(c \times c)$ telle que :

$$\mathbf{H}_{lk} = \sum_{i=1}^{n} \sum_{A_j \neq \emptyset} |A_j|^{\alpha-2} m_{ij}^{\beta} s_{lj} s_{kj} = \sum_{i=1}^{n} \sum_{A_j \supseteq \{\omega_k, \omega_l\}} |A_j|^{\alpha-2} m_{ij}^{\beta} \qquad \forall k, l = \{1, \ldots, c\}. \tag{B.5}$$

Avec ces notations, $\mathbf{V}$ est la solution du système d'équation suivant :

$$\mathbf{HV} = \mathbf{B}. \tag{B.6}$$

Ce système est résolu colonne par colonne (chaque colonne $\mathbf{V}$ est la solution du système linéaire ayant c équations et c inconnues).

Procédure d'optimisation pour l'algorithme EVCLUS

Pour minimiser la fonction objectif de EVCLUS, Denœux et Masson proposent d'utiliser un algorithme de gradient à pas adaptatif dérivé d'une méthode d'apprentissage pour les réseaux de neurones [55, 21]. À l'étape t, il calcule le gradient de chaque paramètre à optimiser afin d'obtenir une direction de déplacement et il établit un pas d'apprentissage. Ce pas est augmenté si la valeur de la fonction objectif a diminué entre deux itérations (la convergence est ainsi plus rapide). Dans le cas contraire, le pas est diminué car il a "sauté" le minimum de la fonction objectif. Des détails sur cette technique sont disponibles dans l'article [21].

Les dérivés partielles de J_{EVCLUS} par rapport aux paramètres a, b et α_{il} $\forall i \in \{1, \ldots, n\}$, $k/A_k \subseteq \Omega$ sont présentés ci-dessous.

$$\frac{\partial J}{\partial a} = \frac{2}{\sum_{i<j} d_{ij}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{K_{ij}(aK_{ij} + b - d_{ij})}{d_{ij}}, \tag{C.1}$$

$$\frac{\partial J}{\partial b} = \frac{2}{\sum_{i<j} d_{ij}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{(aK_{ij} + b - d_{ij})}{d_{ij}}, \tag{C.2}$$

$$\frac{\partial J}{\partial \alpha_{il}} = \frac{2a}{\sum_{i<j} d_{ij}} \sum_{j=i+1}^{n} \frac{(aK_{ij} + b - d_{ij})}{d_{ij}} \frac{\partial K_{ij}}{\partial \alpha_{il}}, \tag{C.3}$$

$$\frac{\partial K_{ij}}{\partial \alpha_{il}} = \sum_{k,k'} \frac{\partial m_{ik}}{\partial \alpha_{il}} m_{jk'} \xi_{kk'}, \quad \text{avec} \quad \xi_{kk'} = \begin{cases} 1 & \text{si } A_k \cap A'_k = \emptyset, \\ 0 & \text{sinon.} \end{cases} \tag{C.4}$$

$$\frac{\partial m_{ik}}{\partial \alpha_{il}} = \begin{cases} m_{ik}(1 - m_{ik}) & \text{si } l = k, \\ -m_{ik} m_{il} & \text{sinon.} \end{cases} \tag{C.5}$$

Comparaison de l'apprentissage actif avec la sélection aléatoire de contraintes pour CECM

Les figures D.1, D.2 et D.3 ci-dessous présentent les résultats de l'algorithme CECM avec $\rho^2 = 1000$ et $\xi = 0.5$ pour les bases de données de l'UCI. La sélection des contraintes est tout d'abord aléatoire et les indices de Rand indiqués par la courbe comprenant des cercles noirs correspondent à des moyennes sur 100 expériences. L'apprentissage actif décrit par l'algorithme 3.2 est ensuite utilisé pour créer la seconde courbe comprenant des croix. Les figures montrent que la qualité de la partition est la plupart du temps meilleure lorsque l'apprentissage actif est utilisé.

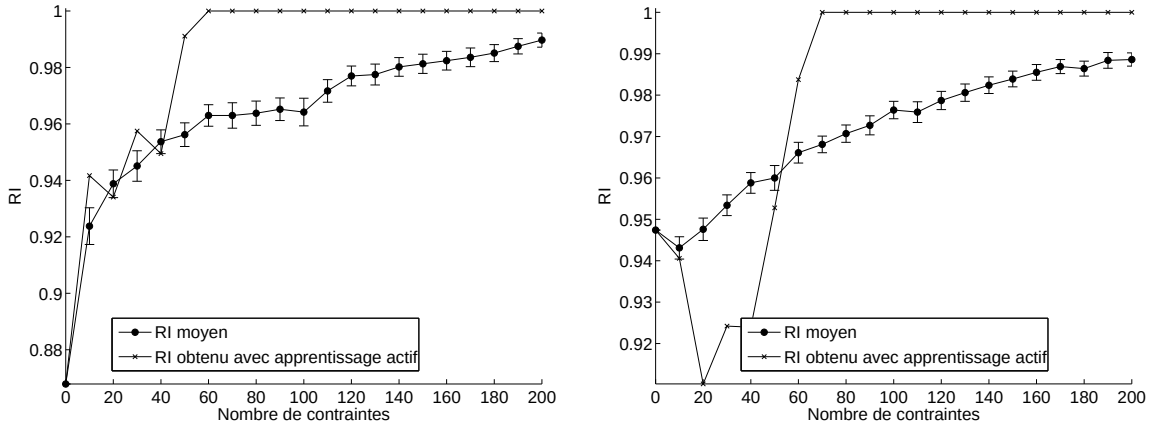


Figure D.1 – Comparaison de l'indice de Rand pour CECM avec apprentissage actif et sélection aléatoire de contraintes avec Iris (a) et Wine (b).

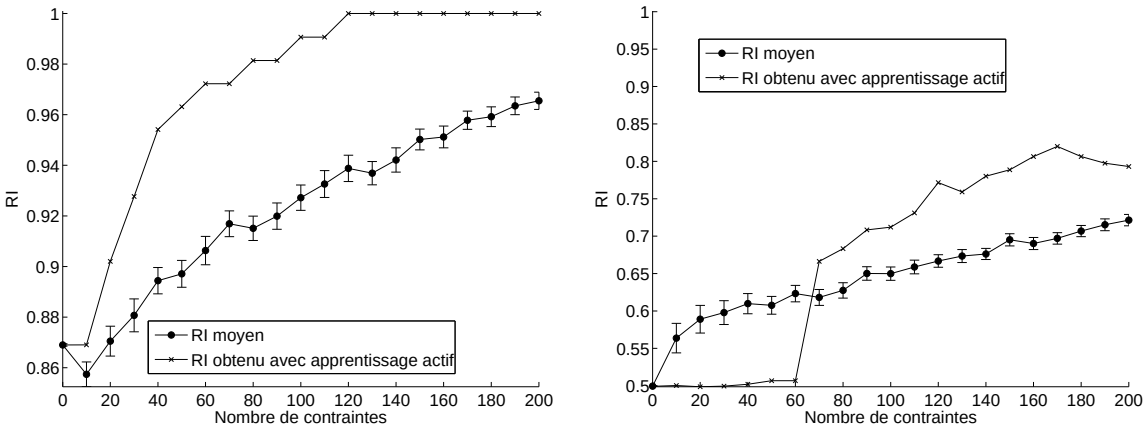


Figure D.2 – Comparaison de l'indice de Rand pour CECM avec apprentissage actif et sélection aléatoire de contraintes avec Glass (a) et Ionosphere (b).

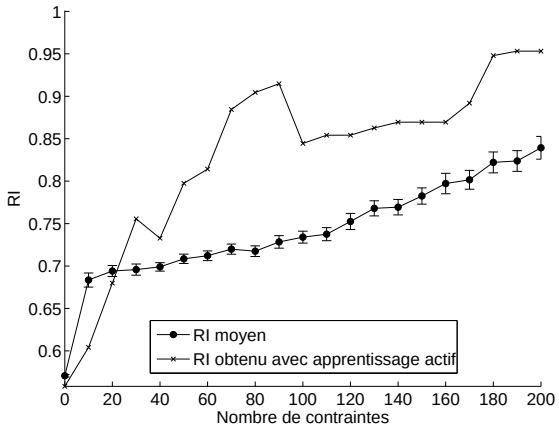


Figure D.3 – Comparaison de l'indice de Rand pour CECM avec apprentissage actif et sélection aléatoire de contraintes avec LettersIJL.