Université Pierre et Marie Curie **UFR d'informatique**

Laboratoire d'informatique de Paris 6 **École doctorale EDITE**

Caractérisation de classes par la découverte automatique de sous-classes

THÈSE

présentée et soutenue publiquement le 23 Septembre 2009

pour l'obtention du

Doctorat de l'université Paris 6

(spécialité informatique)

par

Jason Forest

Composition du jury

Rapporteurs :	Carl FRELICOT (Professeur) Mohammed RAMDANI (Professeur)
Examinateurs :	Matthieu CORD (Professeur) Julien VELCIN (Maitre de conférence)
Directrice :	Bernadette BOUCHON-MEUNIER (Directeur de recherche)
Encadrante :	Maria RIFQI (Maitre de conférence)

Mis en page avec la classe thloria.

Résumé

La fouille de données a pour objectif l'extraction de connaissances à partir de grandes bases de données. La caractérisation de données s'inscrit dans ce domaine et regroupe les méthodes qui mettent en avant les caractéristiques et les tendances des données. Nos travaux se placent dans le cadre de la caractérisation de données en apprentissage supervisées et se focalisent sur la caractérisation de classe.

Dans les jeux de données réelles, il arrive qu'une classe regroupe en son sein plusieurs comportements distincts. On parle alors de *classe non-homogène*. Ce type de classes pose des problèmes aux méthodes de caractérisation usuelles, comme la construction de prototypes ou de résumés, qui ne sont pas en mesure de détecter les différentes tendances. Pour être correctement prises en compte, les classes non-homogènes nécessitent d'être segmentées en sous-classes pertinentes.

Nous proposons, dans notre thèse, une approche novatrice de segmentation automatique de classes en sous-classes qui détecte et isole les différents comportements. Nous proposons également une nouvelle définition du calcul des scores de typicalité qui se base sur ce découpage. Contrairement aux méthodes existantes, qui utilisent un algorithme de clustering, notre approche s'appuie sur l'organisation des classes dans le jeu de données et détecte automatiquement le bon nombre de sous-classes.

Notre algorithme permet une amélioration des méthodes de construction de prototypes flous et donne des résultats de caractérisation plus riches. Notre approche a été testée et validée aussi bien sur des jeux de données artificielles que sur des jeux de données réelles issus, par exemple, du marketing pharmaceutique ou encore des bases d'images.

Mots-clés: sous-classes, caractérisation, prototypes, typicalité.

Table des matières

Introduction

Cadre de la thèse

Cette Thèse a été réalisée dans le cadre d'une Convention Industrielle de Formation par la Recherche (CIFRE) entre le Laboratoire d'Informatique de Paris 6 et l'entreprise ARVEM France S.A. ARVEM est une entreprise de marketing pharmaceutique. Son activité est de démarcher les médecins et de leur faire la promotion de médicaments. Dans le but d'optimiser les visites aux médecins, ARVEM a fait appel au LIP6 pour mettre en place une méthode de traitement de données.

Motivations

L'être humain, grâce à sa maîtrise du langage et des mathématiques, est capable de mesurer, quantifier et ranger dans des catégories. Ainsi, tous les objets qui nous entourent font partie de groupes et sont décrits sous la forme de nombre ou de mots. Beaucoup d'espèces vivantes sont répertoriées, mesurées et disséquées ; chaque chose est étudiée, évaluée et étiquetée. Or toutes ces données ne sont pas amassées dans un unique but de stockage. L'étape suivante de la collecte est l'analyse des données afin d'en tirer le plus d'informations utiles : le plus de connaissance.

L'informatique, et plus particulièrement le domaine de la fouille de données, s'inscrivent dans cette optique. L'objectif est de concevoir des méthodes qui permettent de tirer de la connaissance de très grandes bases de données. Plusieurs tâches peuvent être mises en œuvre selon que les données sont rangées en catégories ou non.

- Dans le cas de données non supervisées, c'est-à-dire qui ne sont pas organisées en classes, des regroupements entre exemples similaires sont recherchés grâce aux méthodes de *clustering*. La connaissance du nombre de groupes d'un jeu de données, par exemple, est une information importante.
- Si les données sont réparties en classes, le but va généralement être d'apprendre une généralisation de cette répartition et de proposer une modélisation du phénomène.

Ce modèle sera utilisé pour classer de nouvelles données non étiquetées.

En marge de ces deux tâches, il existe un domaine moins exploité : celui de la *caractérisation de classes*. A l'instar de l'apprentissage, la caractérisation de classes travaille sur des données supervisées. Cependant, l'objectif n'est pas d'apprendre pour classer de nouvelles données mais de se concentrer sur les données existantes et d'en découvrir les caractéristiques. Les méthodes de résumé de données entrent dans ce cadre.

Dans les jeux de données réelles, il arrive qu'une classe regroupe en son sein plusieurs comportements distincts. On parle alors de *classe non-homogène*. On peut prendre comme exemple la classe des personnes qui n'exercent pas d'activité professionnelle. Ces individus sont soit jeunes, soit âgés, soit au chômage. Ils sont pourtant tous réunis sous la même étiquette bien qu'étant très différents. Ce type de classe est problématique en fouille de données et les méthodes de caractérisation traditionnelles ne sont pas capables de prendre en compte leur richesse et leur complexité.

Nos travaux de recherches se focalisent sur l'étude des classes non-homogènes. Nous pensons que les méthodes existantes de détection et de prise en compte de ces structures n'exploitent pas toutes les informations disponibles.

De plus, un autre phénomène doit être pris en compte en caractérisation de données : *le chevauchement des classes*. Quand les classes d'un jeu de données se mélangent, on obtient des zones de l'espace des données ambigües. Or, les méthodes existantes de segmentation de classes ne les identifient pas et cela dégrade les résultats.

La méthode de caractérisation que nous proposons tient compte du phénomène de classes chevauchantes en identifiant et en filtrant les exemples présents dans ces zones.

Organisation de la Thèse

Ce manuscrit est organisé en trois parties :

La première se focalise sur les méthodes de caractérisation de classes qui existent dans la littérature. Nous commençons par présenter dans le chapitre 1 les approches utilisées en marketing pour extraire les caractéristiques des clients. Puis, en nous plaçant dans un cadre plus général, nous détaillons les méthodes de construction de résumés linguistiques et de prototypes qui permettent de mettre en valeur les tendances des jeux de données. Dans le chapitre 2 nous expliquons que ces méthodes ne permettent pas de traiter correctement les classes non-homogènes et nécessitent une segmentation des classes. Nous faisons, en fin de partie, chapitre 3, un état de l'art commenté des méthodes de segmen-

tation de classes.

La seconde partie de notre manuscrit présente notre contribution au domaine de la segmentation de classes au travers d'un algorithme original. Nous en détaillons le fonctionnement avant d'en présenter les avantages sous la forme d'illustrations sur des jeux de données artificielles en deux dimensions.

La troisième partie est constituée de plusieurs expérimentations et mises en situation de notre méthode ; aussi bien sur des données artificielles, chapitre 1, que sur des données réelles, chapitre 2. Nous montrons notamment que notre approche permet d'aider à la caractérisation de données, que ce soit celles de l'entreprise ARVEM ou bien des données issues d'autres domaines. Dans le dernier chapitre, nous comparons notre méthode de segmentation à une approche à base de classification non supervisée.

Première partie

Caractérisation de classes et de sous-classes

Introduction : Définition et intérêts de la caractérisation des données

Notre thèse traite de la caractérisation de données. Notre premier objectif va être de définir clairement ce que nous entendons par ces termes.

La définition du mot *caractérisation* par le Centre National de Ressources Textuelles et Lexicales (CNRTL, 2009) est la suivante : *« Processus analytique aboutissant à une définition puis à une classification des éléments d'un ensemble. »* Le verbe *caractériser* admet quant à lui cette définition : *« Mettre en évidence le ou les traits dominants ou distinctifs d'une chose ou d'une personne. »* De plus, sous sa forme pronominale le verbe caractériser s'entend comme *« constituer le trait distinctif de quelque chose ou quelqu'un. »* Sur un autre dictionnaire accessible en ligne, le MediaDico (MediaDICO, 2009), on trouve les synonymes du mot *caractérisation* : *spécification, détermination, délimitation, définition, limitation, localisation, indication, datation, repère.*

La caractérisation est donc le fait de définir, de donner une représentation, d'un objet par le biais de ses **caractéristiques**. Par conséquent, *caractériser* un objet suppose de bien connaître sa nature, de savoir ce qui le distingue ou le rapproche d'autres entités et quels en sont les traits significatifs et les limites.

Si on se replace à présent dans le contexte de la caractérisation *de données*, on peut en préciser la définition : la caractérisation de données est donc la découverte et la mise en avant des caractéristiques propres à un jeu de données. Cela implique d'utiliser des méthodes d'analyse des valeurs, du sens et de la structure des données, pour détecter les spécificités de celles-ci en vue d'une utilisation et/ou d'une interprétation plus pertinente.

Le processus de fouille de données se décompose en plusieurs étapes (Fayyad, 1996). La caractérisation de données est une des premières étapes de la phase de découverte de connaissance. En effet, en caractérisant des données, on exhibe la matière première sur laquelle l'analyse portera. La caractérisation ne permet pas de tirer immédiatement

des conclusions ou des interprétations mais elle attire l'œil sur les tendances, souligne les points importants et met en exergue les spécificités. C'est, en somme, le moyen de se concentrer sur ce qui est essentiel et de ne pas se laisser perturber par ce qui n'apporte pas d'information pertinente.

Un point sur lequel nous voulons insister est que *la caractérisation* se distingue de *la classification* de données dans le sens où elle fournit des critères qui permettront par exemple, de ranger ultérieurement les objets en différentes catégories ou bien d'apprendre un bon modèle. La classification fournit des catégories pour des objets inconnus, la caractérisation fournit de l'information. Pour toutes ces raisons, la caractérisation s'inscrit comme une des étapes très importantes du processus de fouille de données (datamining).

Dans ce chapitre nous présentons un état de l'art de la caractérisation de classes, divisé en quatre parties :

Dans le but de répondre à la problématique d'optimisation de la visite des médecins de l'entreprise ARVEM, la première partie se concentre sur les indicateurs et les méthodes utilisés en économie pour caractériser des clients. L'accent est mis sur la modélisation par les valeurs *RFM*.

Puis, dans une seconde partie, nous nous plaçons dans un cadre plus général pour présenter les méthodes de caractérisation de classes qui construisent des résumés linguistiques ainsi que les approches de construction de prototypes à base de typicalité.

La troisième partie introduit la problématique centrale de nos travaux : les classes non-homogènes. Nous présentons plus en détail ce type de classes puis nous expliquons en quoi elles posent un problème pour les méthodes de caractérisation.

En effet, pour être correctement prises en compte, ces classes nécessitent d'être découpées en sous-classes. La dernière partie présente un état de l'art des deux familles d'algorithmes de segmentation de classes. Nous mettons l'accent sur les améliorations qu'il est possible de leur apporter.

Chapitre 1

Méthodes de caractérisation de classes

1.1 Caractérisation en marketing direct

Notre thèse s'est déroulée dans le cadre d'une convention CIFRE. Elle a donc vocation a être appliquée. La société avec laquelle nous avons effectué nos recherches a pour problématique l'optimisation de la présentation de produits pharmaceutiques à des médecins : un certain nombre de commerciaux spécialisés, les délégués à l'information médicale, vont rencontrer les médecins sur leur lieu de travail pour leur faire la présentation d'un panel de médicaments. Le but est bien sûr commercial, mais cette activité permet également aux praticiens de dialoguer avec les laboratoires.

L'action de démarcher directement les personnes physiquement ou par le biais de courrier est connue en économie sous le nom de marketing direct. Pour répondre le mieux possible aux attentes de l'entreprise qui nous encadrait, nous avons effectué un état de l'art des travaux de recherche en traitement de données sur le sujet. Nous avons ainsi pu nous rendre compte que les chercheurs en marketing direct se focalisent sur deux problèmes : quelles données sont importantes et quelles méthodes doit-on utiliser. Pour illustrer ces choix, les recherches se basent sur des situations concrètes comme l'estimation de réponses positives lors d'une campagne de publicité par email (Ling et Li (1998), Eiben et al. (1996)) ou la prédiction de dons à une œuvre de charité (Sousa et al. (2002), Pijls et al. (2001), Potharst et al. (2002), Madeira et Sousa (2002)).

1.1.1 Les données importantes en marketing direct

Le premier problème du marketing direct est le choix des données à utiliser. Dans ce milieu industriel, les données coûtent chères à obtenir et à traiter. Les résultats que l'on peut attendre dépendent notamment des attributs que l'on a sélectionnés. Ainsi, choisir

les bons attributs et le bon nombre est un problème récurrent.

Dans les cas pratiques qui apparaissent dans la littérature, deux grands types de descripteurs se dégagent. On distingue tout d'abord les **descripteurs socio-économiques**. Ceux-ci renseignent sur l'intégration des individus dans leur environnement ou bien les décrivent quantitativement ou qualitativement. Des informations comme l'âge, le sexe, le nombre d'enfant(s), le salaire sont des bons exemples de ce que peuvent être les attributs socio-économiques pour des individus. Le nombre de salariés, le nombre d'années d'existence ou encore le chiffre d'affaires décrivent de la même façon les entreprises clientes.

Un second type d'attributs se retrouve pour décrire un autre aspect des clients potentiels. Ce sont ceux qui explicitent les **comportements d'achats**. Ils relient le consommateur au produit qui va éventuellement lui être proposé. Ce sont souvent les informations les moins chères à obtenir puisqu'elles appartiennent à l'entreprise elle-même (Hughes, 2003).

Mis à part l'article de Wheaton (1996), il existe un quasi consensus sur le meilleur compromis entre choix des attributs et pouvoir de prédiction du modèle obtenu. En effet, la majorité des articles utilisent une modélisation par les **valeurs RFM** (*RFM values*) pour décrire les individus (Madeira & Sousa, 2002; Kaymak, 2001; Sousa et al., 2002; Poel, 2003; Potharst et al., 2002; Sellers & Hughes, 2003; Putten, 1999; Hughes, 2003). Cette approche est utilisée depuis plus de quarante ans (Hughes, 2003).

Les *RFM values* sont trois indicateurs des achats précédents de l'individu.
R : *Recency.* Le nombre de jours écoulés depuis le dernier achat ou la dernière réponse positive à une sollicitation commerciale.
F : *Frequency.* Le nombre de fois où l'individu a répondu positivement à une offre.
M : *Monetary.* La somme totale dépensée dans des offres commerciales.

Mais il faut tout de même relativiser : même si les modélisations basées sur les valeurs RFM semblent concluantes, on ne peut pas prétendre faire un bon ciblage sans prendre en compte certaines variables socio-économiques (Putten, 1999). L'article de Poel (2003) met par exemple en avant l'importance de la variable « *possède un crédit.* »

Donc, comme le souligne Hughes (2003), les valeurs RFM sont importantes mais pas suffisantes.

1.1.2 Technique naïve de modélisation par les valeurs RFM

La méthode naïve de modélisation par les valeurs RFM est utilisée par certaines entreprises pour classer simplement et rapidement les clients. Remarquons que cette méthode se passe d'échantillon d'apprentissage et qu'elle n'utilise que les trois valeurs RFM.

Cette méthode s'appuie sur l'hypothèse que les personnes les plus susceptibles d'acheter sont celles qui ont acheté il n'y a pas longtemps (R), qui achètent fréquemment (F) et pour un montant élevé (M).

On commence par trier les clients par ordre décroissant du nombre de jours depuis le dernier achat (valeur R). On découpe notre liste triée en cinq parties égales. Les individus de la première partie se voient attribuer le numéro un, ceux de la deuxième partie le numéro deux etc. Pour chaque partie on trie les individus par ordre décroissant de fréquence (valeur F) et, une fois encore on découpe en cinq. De la même façon on attribue la valeur de la partie dans laquelle l'individu se trouve. Enfin, on trie les individus par ordre décroissant de montant total d'achat et on découpe en quintiles numérotés (valeur M).

On obtient 125 groupes d'individus notés de 555 à 111. Les individus les plus susceptibles d'acheter et à qui il faut s'adresser en priorité sont ceux du groupe étiqueté 555. Ceux juste ensuite sont ceux du groupe 554, etc. Cette granularité est suffisamment grande pour permettre de choisir le pourcentage de notre population de départ qui doit recevoir la publicité.

Cette méthode est très rudimentaire. Cependant, elle est réputée pour être la plus rapide et la plus accessible (Hughes, 2003).

1.1.3 Techniques d'Intelligence Artificielle

On remarque que deux grandes familles de méthodes se rencontrent dans la littérature :

Il y a tout d'abord la communauté des économistes qui utilise des méthodes issues des statistiques et de l'analyse de données. On trouve souvent des modèles se basant sur de la régression linéaire ou de la régression logistique (Marsh, 2001). Certains auteurs se tournent vers des modèles tels que l'estimation de paramètres grâce aux fonctions de pertes symétriques ou asymétriques (Bult, 1993).

La deuxième famille de méthodes est celle qui nous vient de la communauté de chercheurs en Intelligence Artificielle. On peut trouver des réseaux neuronaux (Zahavi & Levin, 1997; Potharst et al., 2002), utiles pour leur capacité à s'adapter à la non-linéarité des problèmes.

Il existe également des modèles basés sur les arbres de décision (CART et CHAID) et les règles d'association (Haughton & Oulabi, 1993; Madeira & Sousa, 2002) qui permettent une interprétation aisée des résultats même par les non spécialistes.

Les méthodes introduisant la logique floue sont également nombreuses puisque les problèmes rencontrés sont souvent chargés d'incertitudes et/ou d'imprécisions (Kaymak,

2001; Kaymak & Setnes, 2000; Sousa et al., 2002).

Enfin, les méthodes évolutionnaires comme les algorithmes génétiques (Bhattacharyya, 1999) ou la programmation génétique (Eiben et al., 1996) sont moins utilisées bien qu'elles permettent de trouver des solutions originales.

Il existe beaucoup d'articles qui font des tests comparatifs entre les différentes méthodes (par exemple (Sousa et al., 2002), (Madeira & Sousa, 2002) ou encore (Eiben et al., 1996)). En comparant les taux de bonnes prédictions, la conclusion est que les méthodes se valent toutes, globalement. Cependant, certains auteurs comme Madeira et Sousa (2002) ont montré que les méthodes à base de modèles flous donnent des résultats plus stables.

1.1.4 Problèmes fréquents

Le premier problème qui peut surgir en marketing direct est le déséquilibre des classes dans l'échantillon d'apprentissage comme le souligne Ling et Li (1998). Cela vient du fait que le taux de réponses positives à une sollicitation publicitaire est bas, souvent en dessous de 5%. Ainsi les méthodes d'apprentissage à employer doivent pouvoir prendre en compte des classes déséquilibrées. On pense notamment aux méthodes de *bagging* ou de *boosting* qui peuvent prendre des sous-bases d'apprentissage artificiellement équilibrées.

Le second problème vient du fait que beaucoup de méthodes d'apprentissage donnent une classification des individus sans permettre de déterminer quel est le pourcentage de chances qu'un individu réponde favorablement. Pour corriger cela, on peut modifier les algorithmes pour qu'ils fournissent un certain degré de fiabilité dans les résultats.

Enfin, le troisième problème est celui de la taille des données à traiter. Beaucoup d'entreprises ont des fichiers clients très volumineux. Une fois encore, le choix de l'algorithme sur ce critère est décisif.

1.1.5 Synthèse

Ces travaux ont une utilité pour la phase de collecte des données. Nous nous en sommes servis lors de la constitution de notre base de données (que nous détaillons dans la partie expérimentations).

Cependant, on s'aperçoit que ce qui est fait dans le domaine du marketing direct, quant à la caractérisation de données, est plutôt pauvre : Résumer les individus par trois valeurs n'est pas une solution envisageable si on cherche à extraire toute l'information qui est contenue dans une base de données.

Dans cette optique, nous avons décidé de réorienter nos travaux et de nous placer dans une problématique plus générale de *caractérisation des classes* ; qu'elles soient marketing, médicales, sur des images ou bien abstraites.

1.2 Méthodes de résumé

Dans la famille des méthodes de caractérisation des classes, les méthodes de résumé ont une place de choix. Elles permettent en effet d'extraire sous une forme intelligible (souvent linguistique) la substance des données. Les méthodes que nous présentons ici se placent toutes dans le cadre de la logique floue, car c'est dans ce domaine qu'il y a le plus grand nombre de travaux.

1.2.1 Résumés linguistiques : extraction de règles floues

Parmi les méthodes de résumé d'information il en existe certaines qui permettent de décrire les données linguistiquement. Ces méthodes sont particulièrement intéressantes car les résumés qu'elles produisent peuvent être analysés par des non experts en logique floue. On obtient en effet une caractérisation parfaitement intelligible et très simple d'accès.

Le formalisme qui est utilisé est issu de la théorie des sous-ensembles flous développée par Zadeh (1965). Avant de présenter les méthodes de résumé qui utilisent ce formalisme nous allons faire un bref rappel de la théorie des sous-ensembles flous. Puis, nous nous intéresserons aux variables linguistiques qui font le lien entre les données et les étiquettes linguistiques. Ensuite, nous présenterons un panorama des méthodes de résumé et d'extraction de connaissances basées sur la logique floue.

1.2.2 Variables et quantificateurs linguistiques

Les méthodes de résumé de données qui utilisent les sous-ensembles flous s'appuient sur deux concepts particulièrement importants : les *variables linguistiques* et les *quantifieurs linguistiques*.

Les variables linguistiques ont été introduites par Zadeh (1975). Il s'agit de variables qui admettent pour valeur non pas un nombre mais un mot ou une phrase. L'utilisation d'un mot ou d'une phrase permet d'être moins spécifique qu'avec un nombre.

Plus formellement, les variable linguistiques sont définies par Bouchon-Meunier (2007) comme un triplet (V, D, T_V) où : V est une variable (température, âge, ...) qui est définie sur le domaine D (par exemple $[0, 120]$ pour l'âge et $[-50, 50]$ pour la température) ; T_V

est un ensemble de sous-ensembles flous décrivant V grâce à leur fonction d'appartenance μ_V et étiquetée par un terme linguistique (par exemple « froid »pour la température.)

La figure 1.1 représente la variable linguistique « Température. » Celle-ci est définie par : $V =$ température, $D = [0, 50]$ et $T_V = \{$froid, tiède, chaud$\}$.

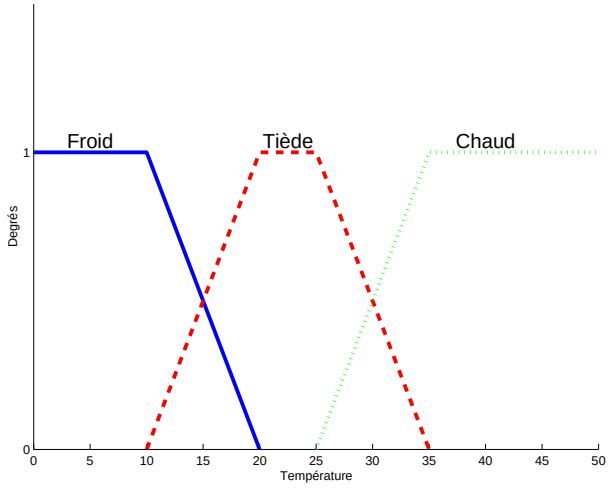

FIGURE 1.1 – Variable linguistique "Température".

Un autre outil très important pour le résumé linguistique est le *quantificateur flou*. Un quantificateur flou ressemble à une variable linguistique à la différence qu'au lieu de renseigner sur une valeur floue il se focalise sur une quantité floue.

Les quantificateurs flous ont été formalisés par Zadeh (1983). Leur but est de modéliser les quantificateurs linguistiques tels que : "la plupart", "généralement", "rarement", etc. Formellement, (Bouchon-Meunier, 2007) un quantificateur flou est un sous-ensemble flou de l'ensemble $[0, 1]$ qui décrit un nombre de cas ou une proportion approximative.

1.2.3 Méthodes de résumés linguistiques de Yager

Les premières références sur l'utilisation des sous-ensembles flous pour construire des résumés linguistiques se trouvent dans les travaux de Yager et Rubinson (1981) et Yager (1982). Les auteurs proposent de fournir des assertions de la forme : « Q éléments de D sont a pour A »où Q est un quantificateur flou tel que l'a défini Zadeh (1983), D est l'ensemble de référence et a est une valeur de la variable linguistique A. Par exemple l'assertion « la plupart des employés sont d'âge moyen »admet Q =la plupart comme quantificateur, D est l'ensemble des employés et a est la valeur « *moyen* »pour la variable linguistique A =âge.

Etant donné un ensemble de quantificateurs flous et de variables linguistiques, on peut

construire plusieurs assertions qui représenteront la situation avec des degrés différents de vérité. Afin de trouver les assertions qui ont le plus de pertinence, Yager (1982) propose d'associer à chacune un degré de validité τ calculé comme suit : si on considère l'assertion « *Q éléments de D sont a pour A* », pour chaque élément d de D on peut calculer son degré d'appartenance $\mu_a(d)$ à la valeur a. Ainsi, pour l'ensemble des individus de la base de données on peut calculer l'agrégation des degrés d'appartenance R_a :

$$R_a = \frac{1}{|D|} \sum_{d \in D} \mu_a(d)$$

Il ne reste plus qu'à appliquer le quantificateur flou pour obtenir le degré de validité. On va donc calculer le degré d'appartenance de R_a à la fonction μ_Q associée à Q.

$$\tau = \mu_Q(R_a)$$

L'indice τ permet de savoir à quel point le résumé reflète la réalité. Grâce à τ on peut classer les assertions par ordre décroissant de validité.

Cette approche a été étendue dans Yager (1991) pour prendre en compte les résumés linguistiques conjonctifs (« *Q éléments de D sont a pour A ET b pour B* ») et disjonctifs (« *Q éléments de D sont a pour A OU b pour B* »). La même logique est utilisée, à part que l'agrégation tient compte de plusieurs variables linguistiques :

- Dans le cas d'une conjonction, les fonctions d'appartenance sont calculées à l'aide d'une norme triangulaire $\top$:

$$R_{a \top b} = \frac{1}{|D|} \sum_{d \in D} \top(\mu_a(d), \mu_b(d))$$

- Dans le cas d'une disjonction, les fonctions d'appartenance sont calculées à l'aide d'une conorme triangulaire $\perp$:

$$R_{a \perp b} = \frac{1}{|D|} \sum_{d \in D} \perp(\mu_a(d), \mu_b(d))$$

Ces approches se focalisent sur l'ensemble de la base de données. Or, on peut vouloir ne s'intéresser qu'à une certaine sous-population. Pour cela, on peut utiliser l'approche suggérée par Yager (1991) et qui génère des assertions du type : « *les Q éléments de D qui sont a pour A sont b pour B.* » Ce type de résumé est intéressant dans le sens où il permet de se focaliser uniquement sur un petit groupe d'éléments qui remplissent une certaine condition.

Le calcul qui permet d'obtenir le degré de validité d'une assertion, devient dans ce cas :

$$\tau_{a,b} = \mu_Q(R_{a,b}) \text{ où } R_{a,b} = \frac{\sum_{d \in D} \top(\mu_a(d), \mu_b(d))}{\sum_{d \in D} \mu_a(d)}$$

On notera qu'il est possible de combiner les résumés conjonctifs, disjonctifs et ceux portant sur une partie de la population de la base pour construire des résumés complexes et plus spécifiques. Pratiquement parlant, à partir d'un nombre relativement grand d'éléments dans la base de données il est possible d'obtenir beaucoup de résumés. Cela est dû au nombre de combinaisons possible.

L'approche de construction des résumés linguistiques de Yager a donné lieu à de nombreuses extensions. Par exemple, on trouve des moyens alternatifs pour calculer le degré de validité τ tel que les opérateurs d'agrégation *OWA* définis par Yager (1988) que nous détaillons dans l'annexe 1.3. On trouve également d'autres moyens pour construire des conjonctions et disjonctions avec, par exemple, des règles d'association comme le font Kacprzyk et Zadrozny (2003).

En outre, le nombre de résumés qu'il est possible de construire est si grand que George et Srikanth (1996) ont proposé d'ajouter des heuristiques de découverte de résumés qui maximisent le degré de validité τ grâce à un algorithme génétique.

Ce type de méthodes a été implémenté et développé par Rasmussen et Yager (1997) dans un programme d'interrogation de bases de données par requêtes flexibles du nom de *SUMMARYSQL*. Ce programme fournit une extension au logiciel de base de données Access et permet de calculer le résumé d'une assertion sur une base. L'utilisateur formule une requête en langage de base de données (SQL) et donne un quantificateur flou ainsi qu'une variable linguistique. Le système lui renvoie le degré de vérité associé.

Dans la même famille d'applications, Kacprzyk (1999) a proposé une autre extension du logiciel Access qui, étant donné un quantificateur et des variables linguistiques, construit l'ensemble des résumés linguistiques associés avec leur degré de validité.

1.2.4 Autres méthodes de résumés par la logique floue

Bosc et al. (2000) ont mis en avant un défaut des méthodes de résumés linguistiques de Yager et Rubinson (1981) : « *l'utilisation de cardinalités scalaires fondées sur des sommes de degrés d'appartenance conduit, par exemple, à regarder comme équivalents une collection de degrés peu élevés dont la somme vaut* 1*, et un unique élément dont le degré d'appartenance vaut* 1. »Ces auteurs ont proposés une approche différente du résumé linguistique en utilisant un calcul de cardinalités floues.

Dans sa thèse, Raschia (2001) a développé une autre approche du calcul de résumés (non linguistique) qui utilise la logique floue : le modèle SAINTETIQ. « *Le modèle SAINTETIQ [...] met en œuvre un algorithme d'apprentissage de concepts qui génère de façon incrémentale une hiérarchie de résumés partiellement ordonnés du plus générique au plus*

spécifique. »Cette approche se sert des thésaurus relationnels flous.

1.3 Degré de typicalité et prototypes

1.3.1 Prototypes et Typicalité

Dans cette partie nous allons présenter les prototypes à base de typicalité. Cet outil permet de caractériser l'information en la synthétisant. En effet, un prototype est un objet qui est très représentatif de sa catégorie. Ainsi, en découvrant, pour chaque catégorie, le prototype qui lui est associé, on obtient une vision très synthétique des différentes classes de la base de données.

La notion de prototype repose sur la notion de typicalité qui, elle même, s'appuie sur les concepts de similarité et de dissimilarité. Il s'agit de comparer les objets les uns par rapport aux autres et de quantifier à quel point ils se ressemblent ou non.

Dans la partie suivante, nous commencerons par définir formellement un prototype. Puis, nous nous intéresserons aux méthodes de construction de prototypes qui utilisent la typicalité.

Définitions

Un **prototype** est un objet qui est choisi pour donner une représentation simple et compréhensible de son groupe, c'est-à-dire qu'il résume les caractéristiques principales de sa catégorie.

Le concept de prototype repose sur la notion de *typicalité*. En effet, tous les objets d'un même groupe n'ont pas la même représentativité ; certains sont plus typiques et sont donc plus à même d'être pris comme représentants. Par exemple, si on désire montrer un prototype de la famille des oiseaux, on aura tendance à prendre un canard et on s'abstiendra de choisir une autruche car il est admis que la quasi-totalité des oiseaux volent.

On peut noter que deux cas de figure sont envisageables quand on souhaite exhiber un prototype à partir d'une base de données :

- soit le prototype est un objet réel de la base, choisi parmi tous les autres,
- soit il s'agit d'un objet artificiel qui est construit exprès pour jouer le rôle de représentant.

La notion de **typicalité** a été beaucoup étudiée en psychologie. Dans son travail, Rosch (1978) à notamment montré que la typicalité d'un élément, pour une catégorie donnée, dépendait de deux facteurs : sa *ressemblance* avec les autres membres de sa catégorie et sa

différence par rapport aux membres des autres catégories. En effet, un objet est typique s'il partage beaucoup de caractéristiques communes avec ceux de sa catégorie et s'il se distingue bien des objets des autres catégories.

Il est en effet possible, grâce à la description d'un objet, qu'elle soit numérique ou symbolique, de calculer à quel point il ressemble à un autre en utilisant une *mesure de ressemblance*. De la même façon on peut quantifier son degré de différence grâce à une *mesure de dissimilarité*. Les mesures de ressemblance et de dissimilarité rentrent dans la famille des mesures de comparaison.

Rifqi (1996a); Rifqi (1996b) a proposé de distinguer trois aspects de comparaisons qui mènent ainsi à trois classes de mesures : la *satisfiabilité* compare un objet à un élément de référence, *l'inclusion* quant à elle compare des objets de haut niveau d'abstraction, enfin, la *ressemblance* compare deux objets de même nature sans que l'un d'eux ne soit pris en tant que référence. Les mesures de comparaison travaillent sur les caractéristiques des objets. Par exemple, si on souhaite comparer deux objets x et y du domaine D, de caractéristiques respectives X et Y, on se focalisera sur les éléments distinctifs, $X - Y$ et $Y - X$ et sur les éléments communs, $X \cap Y$.

Les mesures de ressemblance ont été formalisées par Rifqi (1996a) et Rifqi (1996b) comme une relation r qui vérifie :

- r est une fonction normalisée

$$\begin{aligned} r : D \times D &\rightarrow [0,1] \\ (x,y) &\mapsto r(x,y) = F(X \cap Y, X - Y, Y - X) \end{aligned}$$

  où D est l'espace des données et F une fonction à trois arguments.
- r est croissante avec les éléments communs de x et y, $X \cap Y$,
- r est décroissante avec les éléments distinctifs de x et y, $X - Y$ et $Y - X$,
- r vaut 0 si $X \cap Y = \emptyset$. Autrement dit $F(\emptyset, \cdot, \cdot) = 0$,
- r vaut 1 si il n'y a pas d'élément distinctifs. c'est-à-dire : $F(\cdot, \emptyset, \emptyset) = 1$,
- F vérifie que $F(u, v, w) = F(u, w, v)$.

Rifqi (1996a) a également formalisé les mesures de dissimilarité comme une relation d vérifiant :

- d est une fonction normalisée

$$\begin{aligned} d : D \times D &\rightarrow [0,1] \\ (x,y) &\mapsto d(x,y) = G(X - Y, Y - X) \end{aligned}$$

- d est indépendante des éléments communs de x et y,
- r est croissante avec les éléments distinctifs de x et y,
- r vaut 0 si il n'y a pas d'élément distinctifs ; $F(\emptyset, \emptyset) = 0$.

Lesot (2005b) a proposé une extension de ces travaux pour prendre en compte des données numériques de dimension d. Elle a proposé de considérer une mesure de ressemblance comme une fonction se basant sur une mesure de distance ou sur le produit scalaire. La formalisation est alors simplifiée : une mesure de ressemblance se définit comme une fonction r :

$$\mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1]$$

Où r est décroissante en fonction de la distance ou croissante en fonction du produit scalaire. De la même façon, une mesure de dissimilarité est une fonction à valeurs dans $[0, 1]$, croissante en fonction de la distance ou décroissante en fonction du produit scalaire.

Le tableau 1.1 présente quelques exemples de fonctions de ressemblance et de dissimilarité usuelles (Lesot, 2005a).

Méthodes de construction de prototypes par typicalité

Dans Rifqi (1996b), une méthode de construction de prototypes implémentant les travaux de Rosch (1978) sur la typicalité est donnée. Dans celle-ci, l'auteur propose de calculer deux nouvelles notions :

- La *ressemblance interne* d'un objet x de catégorie c_x, qui est une agrégation de tous les scores de ressemblance entre l'objet x et tous les objets y de la même catégorie que lui.

$$\forall y \in D \mid c_y = c_x,\ Ress_{int}(x) = Agreg((r(x, y))$$

- La *dissimilarité externe* d'un objet x, qui est l'agrégation de tous les scores de dissimilarité entre l'objet x et tous les objets z qui ne sont pas de sa catégorie.

$$\forall z \in D \mid c_z \neq c_x,\ Diss_{ext}(x) = Agreg(d(x, z))$$

Les mesures de ressemblance, de dissimilarité et l'opérateur d'agrégation sont autant de choix qui dépendent des données et qui doivent donc être judicieusement sélectionnés. Toujours est-il que ces deux formules permettent de bien rendre compte des deux aspects de la typicalité que Rosch a mis en évidence.

Rifqi (1996b) est allé plus loin en proposant de calculer le score de typicalité d'un objet x comme l'agrégation de son score de ressemblance interne et de dissimilarité externe :

$$Typi(x) = Agreg(Ress_{int}(x), Diss_{ext}(x))$$

Le score de typicalité est donc un nombre appartenant à $[0, 1]$, la valeur 0 signifie que l'objet n'est absolument pas typique de sa catégorie, et la valeur 1 signifie qu'il n'y a pas plus typique, c'est-à-dire pas plus représentatif. On met en valeur les points communs entre individus d'une même catégorie mais également les caractéristiques discriminantes.

La dernière étape est la construction des prototypes. En disposant du score de typicalité de chaque objet de la base de données, le choix des prototypes pour chaque catégorie se fait en sélectionnant le ou les individus qui ont les scores de typicalité les plus élevés. Le prototype est, par conséquent, défini comme l'agrégation des données les plus typiques.

Rifqi (1996a) se place dans un contexte où les objets manipulés sont des sous-ensembles flous. Les prototypes obtenus par cette méthode sont eux aussi des sous-ensembles flous. Lesot (2005b); Lesot (2005a) ont étendu cette approche aux données numériques, c'est-à-dire aux données sous forme de vecteurs appartenant à $\mathbb{R}^p$, pour être en mesure de construire des prototypes flous.

Leur étude se place dans un cadre non supervisé ; les données ne sont pas rangées en catégories. Dans une première étape un algorithme de clustering est utilisé pour découper le jeu de données. Dans un second temps le score de typicalité de tous les objets est calculé en se basant sur des mesures de ressemblance et de dissimilarité adaptées aux données numériques.

L'originalité de ce travail se situe dans la dernière étape : les auteurs ont proposé de définir un prototype comme un ensemble flou dont le noyau contient les points qui ont un score de typicalité supérieur à un certain seuil et dont le support est formé des points dont la typicalité est supérieure à un second seuil plus petit. On obtient ainsi, pour chaque catégorie, un prototype qui la caractérise et la résume.

Mesures de ressemblance R	
produit scalaire	$\langle x, y \rangle$
basée sur une distance normalisée	$1 - \frac{1}{Z} dist(x, y)$
polynomiale	$\left(\frac{\langle x, y \rangle}{Z} + 1 \right)^{\gamma}$
gaussienne	$\exp\left(-\frac{1}{2\sigma^2} dist(x, y)^2 \right)$
Fermi-Dirac (Rifqi et al., 2000)	$z\left(\frac{1}{1 + \exp \frac{dist(x, y) - d_M}{\Gamma}} \right)$
Mesures de dissimilarité D	
distance normalisée	$\frac{1}{Z} dist(x, y)$
dépendant d'un mesure de ressemblance	$1 - R(x, y)$

TABLE 1.1 – Exemples de fonctions de ressemblance et de dissimilarité usuelles entre deux données vectorielles x et y. $dist(x, y)$ est une mesure de distance entre x et y, Z est un coefficient de normalisation, z est est une fonction de normalisation, γ, σ, d_M et Γ sont des paramètres.

Chapitre 2

Le cas particulier des classes non-homogènes

Dans le cadre de l'apprentissage supervisé, c'est-à-dire quand les données sont étiquetées en catégories, le but est généralement d'inférer un modèle prédictif. Pour cela, on s'appuie sur la répartition spatiale des exemples. L'hypothèse qui est faite est que les exemples d'une même classe vont se *ressembler* les uns les autres. Par conséquent, ils seront proches dans l'espace des données. Cependant, cette ressemblance n'est évidemment pas parfaite, et la répartition des exemples donne aux classes des *formes* très variées - car il n'y a aucune garantie qu'une classe soit sphérique.

De plus, il n'y a également *aucune garantie sur le fait qu'une classe est constituée d'un unique regroupement d'exemples.* Cette constatation est lourde de sens car elle sous-entend qu'une classe, c'est-à-dire un ensemble d'exemples rassemblés dans une même catégorie, peut être constituée de plusieurs regroupements. Par conséquent, une même classe peut contenir plusieurs comportements et donc *plusieurs sous-classes.*

Dans la première partie de ce chapitre, nous nous attacherons à définir ce type de classes que nous nommons *classes non homogènes* et nous essaierons de mettre en avant les causes de ce phénomène. Puis, dans une seconde partie, nous verrons que ces catégories non homogènes soulèvent un grand nombre de questions et qu'elles remettent en question l'utilisation des méthodes de caractérisation de classes que nous avons vues précédemment. Enfin, dans une troisième partie, nous expliquerons pourquoi nous pensons que le passage par une sous-catégorisation est une bonne solution à ce problème.

2.1 Définition d'une classe non homogène

La première constatation est qu'il n'existe pas, à notre connaissance, dans la littérature, de distinction formelle entre les classes compactes et les classes non homogènes. Cela s'explique par le fait que les chercheurs du domaine ne font pas l'hypothèse que des données réelles vont se comporter de façon « parfaite »et s'organiser en catégorie compacte. Pourtant il nous semble important de poser la définition de cette différence car elle impacte beaucoup les processus de caractérisation de classes que nous avons vus.

Donc, une classe non homogène est un ensemble d'exemples appartenant à la même catégorie qui, dans l'espace des données, forme *plusieurs regroupements distincts* (ou *clusters*). On peut voir un exemple de classe non homogène dans la figure 2.1 : La classe 1 (croix bleues) est divisée en deux sous-classes.

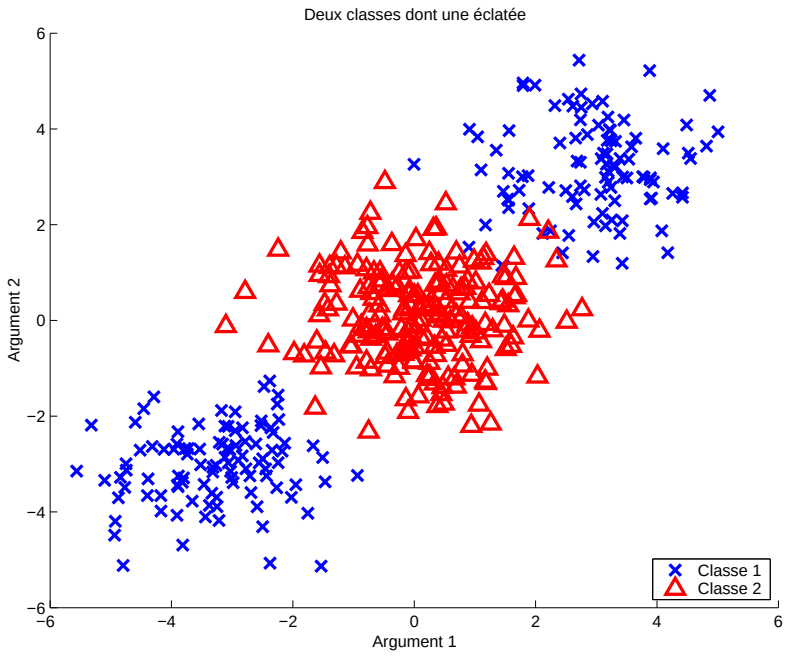

FIGURE 2.1 – Exemple d'une classe non homogène : la classe 1 constituée de croix bleues est formée de deux groupes.

Au niveau de la sémantique, une catégorie non homogène traduit le fait que *plusieurs comportements* sont présents. En effet, les exemples de chacun des groupes de la classe ne se ressemblent pas - ils n'ont pas de points communs, si ce n'est leur étiquette de classe. Il y a donc plusieurs façons d'appartenir à cette classe dans l'espace des caractéristiques données.

Prenons un exemple illustratif : un professeur d'école primaire analyse les moyennes

du trimestre de ses élèves. Il décide (de façon arbitraire) que ceux qui ont une moyenne supérieure ou égale à 14/20 sont rangés dans la catégorie des « bons élèves » alors que les autres sont des « élèves standards ». Souhaitant aller plus loin dans son analyse, il demande aux parents d'élèves combien de temps, en moyenne, est consacré aux devoirs et aux leçons chaque soir de la semaine. En utilisant le nombre d'heures de travail personnel en fonction de la moyenne des élèves il obtient le graphique de la figure 2.2. Il constate alors que la catégorie des bons élèves est divisée en deux groupes : les « bosseurs », qui travaillent beaucoup chez eux et les élèves naturellement doués qui ne fournissent pas une quantité de travail importante et réussissent pourtant bien. La classe des « bon élèves » est donc une classe non homogène, deux comportements distincts y cohabitent.

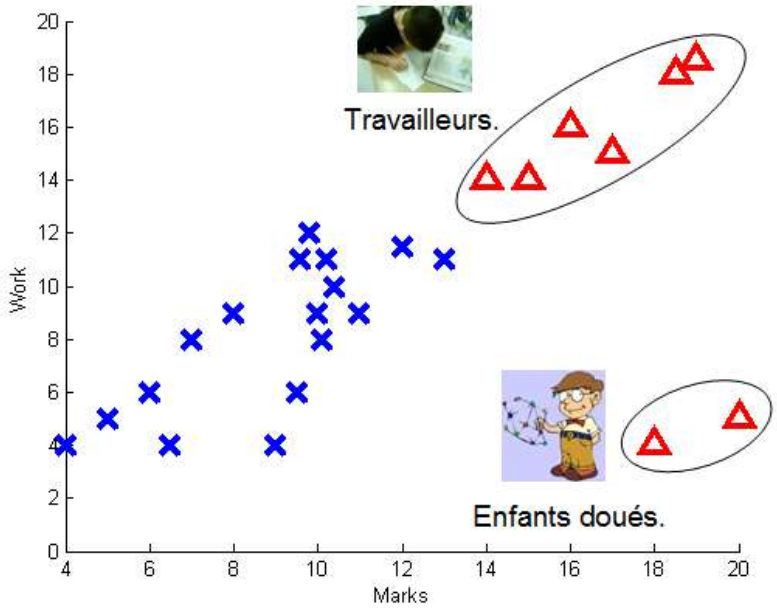

FIGURE 2.2 – Illustration d'une classe non homogène.

Cet exemple est certes assez basique, mais il illustre bien la problématique sous-jacente : plusieurs comportements au sein d'une classe entraînent l'existence de sous-groupes. Nous sommes convaincus que des données issues de situations réelles ont de fortes chances de contenir des classes non homogènes.

On peut se demander à quoi est dû le fait que des classes sont constituées de différents sous-groupes. Les raisons sont multiples :

- La première raison est que les catégories peuvent être à un trop haut niveau sémantique et ne pas correspondre à la réalité « terrain ». Par exemple, un ensemble de photographies étiquetées « Vacances » contiendra à coup sûr plusieurs sous-catégories.
- Ensuite, il se peut que les données soient décrites par trop d'attributs. En effet, cela

ajoute des dimensions qui vont venir séparer les exemples d'une même classe.
- De plus, dans certains jeux de données, les classes sont faites artificiellement et ce n'est qu'après constitution qu'on décide d'utiliser ces données pour faire de la caractérisation ou de l'apprentissage. C'est le cas par exemple des regroupements temporels où toutes les données d'une période de temps sont rangées dans la même catégorie.
- Enfin, la dernière raison est que les données peuvent être bruitées. Le bruit peut alors s'apparenter à une sous-classe.

2.2 Problématiques

Il existe, selon nous, une dichotomie : d'un côté il n'y a pas, dans la littérature, de distinction clairement définie entre des classes homogènes et d'autres non-homogènes ; d'un autre côté la présence de sous-groupes perturbe de façon sensible les différentes méthodes de caractérisation de classe que nous avons vues précédemment.

C'est notamment le cas avec les indicateurs statistiques usuels. Ceux-ci donnent une tendance *globale* alors qu'elle décrit mal la réalité. Cette situation est illustrée par l'exemple évident du calcul de la valeur moyenne de deux groupes de points distants.

Reprenons l'exemple de notre professeur : la valeur moyenne du nombre d'heures passées en travail personnel est un mauvais indicateur : pour la classe des bons élèves, le comportement minoritaire de la classe se retrouve fortement atténué.

Le même phénomène se retrouve dans le calcul de la typicalité. La typicalité se base sur les deux notions de ressemblance interne et de dissimilarité externe. Mais celles-ci ne s'appliquent que si tous les points d'une classe sont proches les uns des autres et loin des autres classes. En présence de classes non homogènes les scores de typicalité sont très faibles empêchant une caractérisation fiable.

Les classes non homogènes posent donc un problème qu'il convient de résoudre. C'est ce que nous proposons de faire dans la suite de notre manuscrit.

2.3 La solution consiste à découper

Comme nous l'avons énoncé, les classes non homogènes possèdent plusieurs comportements qui sont rangés derrière la même étiquette - il y a plusieurs causes pour la même conséquence. La thèse que nous défendons est que le découpage en catégories peut ne pas

suffire et qu'il est donc nécessaire de le raffiner en proposant un niveau supplémentaire : les sous-catégories. Pour cela il convient d'identifier dans les données les différents groupes que forment les exemples. Une fois en possession de ce découpage plus fin, il est possible de le prendre en compte dans les diverses méthodes existantes.

Par conséquent, pour que les méthodes usuelles puissent prendre en compte correctement les classes non homogènes, il est nécessaire de détecter les différents sous-groupes et d'utiliser des méthodes capables d'exploiter cette information.

Chapitre 3

Etat de l'art des méthodes de décomposition de classes

L'idée du découpage de l'information n'est pas récente : dans leur ouvrage, Cormen et al. (1994) transposent à l'algorithmique le célèbre précepte de Machiavel : « diviser pour régner. » Cette approche propose de découper un problème donné en plusieurs sous-problèmes plus simples à résoudre. La combinaison des solutions permet de résoudre le problème global. Ce paradigme est la généralisation du processus de caractérisation de classe par découverte de sous-groupes compacts. Or, la découverte de sous-classes n'est possible que si on décompose de façon judicieuse les classes.

Dans la suite du chapitre, nous nous intéresserons tout d'abord à la formalisation du problème de la décomposition des classes. Puis, nous verrons que lorsqu'il s'agit de découper un jeu de données, deux familles d'algorithmes se distinguent clairement. Ainsi, nous nous concentrerons tout d'abord sur des méthodes de découpage de l'espace dont le but est la classification. Dans un second temps, nous focaliserons notre étude sur les méthodes de segmentation qui servent de prétraitement pour améliorer des méthodes existantes. Nous terminerons par une analyse critique de ces différentes méthodes qui nous amènera à proposer une nouvelle méthode de segmentation.

3.1 Formalisation de la décomposition de classes

C'est dans l'article de Anand et al. (1995) que l'on trouve l'une des premières références à un découpage du jeu de données pour améliorer la classification. En effet, les auteurs proposent la construction de réseaux de neurones modulaires (comme on peut le voir sur la figure 3.1 page 36) pour accélérer les temps de calculs et augmenter le taux de bonne classification.

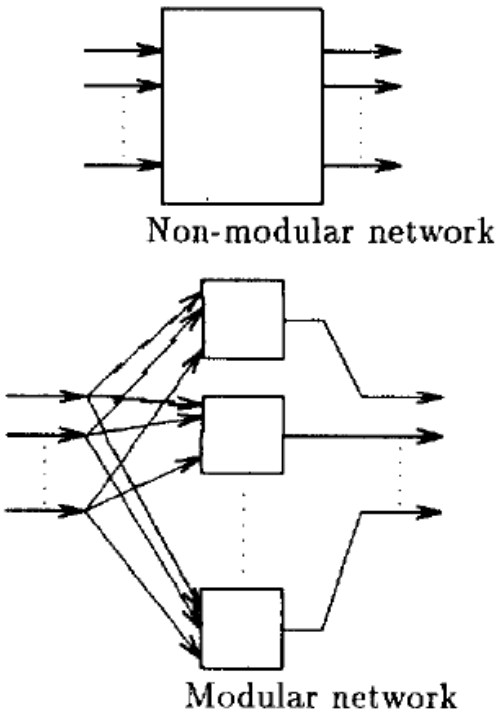

FIGURE 3.1 – Différence entre un réseau de neurones traditionnel et un réseau modulaire.

Ils utilisent une analogie aux différentes zones spécialisées du cerveau pour expliquer l'architecture modulaire de leur modèle : les données sont découpées et chaque "morceau" est donné à un réseau de neurones qui se spécialise dans l'apprentissage de celui-ci. Dans cette approche, les auteurs ne parlent pas de classes non homogènes, mais l'idée de se spécialiser en travaillant sur des sous-classes est esquissée.

Cependant, dans l'article le découpage proposé est assez simple et ne remet pas en cause l'intégrité des classes. En effet, ils proposent de former tous les couples de catégories et de travailler sur chaque couple séparément. Par exemple, un jeu de données formé de 3 classes $\{a, b, c\}$ sera découpé en 3 sous-jeux de données constitués des classes : $\{a, b\}$, $\{b, c\}$ et $\{a, c\}$. D'un problème à K classes, on passe à K problèmes à deux classes.

On notera que cette méthode est évaluée et validée empiriquement sur plusieurs jeux de données réelles.

Quelques années plus tard, Lu et Ito (1999) décident de formaliser la *décomposition*. Ils en distinguent trois types :

– La décomposition explicite où un expert du domaine décide comment les données doivent être découpées pour que chaque module du classifieur se spécifie correcte-

ment.

- La décomposition de classe : le jeu de données est décomposé en sous-jeux de données en fonction des différentes classes. Les réseaux de neurones modulaires rentrent dans cette catégorie car chaque module prend en entrée un sous-jeu composé de deux classes.
- Enfin, la « décomposition automatique »(Gallinari, 1998). Après quelques itérations de la phase d'apprentissage on décompose le problème. Les méthodes de mélanges d'experts rentrent dans cette catégorie.

C'est la deuxième catégorie qui nous intéresse plus particulièrement. L'article de Guan et Zhu (2004) va plus loin dans cette formalisation de la décomposition. Les auteurs proposent de définir les problèmes de classification sous la forme :

$$f : X \rightarrow T$$

f est la fonction qui, étant donné un exemple x de l'ensemble des exemples X, renvoie t une classe de T, l'ensemble de classes. Bien sûr, f doit optimiser le taux de reconnaissance sur l'échantillon d'apprentissage ξ des p exemples du jeu d'apprentissage :

$$\xi = \{(X_i, T_i)\}_{i=1}^{p}$$

Dans ce cadre, faire une décomposition du problème revient alors à décomposer T l'ensemble des classes :

$$T = T^1 \bigcup T^2 \bigcup \ldots \bigcup T^r$$

Et la résolution du problème se transforme donc en la découverte de toutes les fonctions f_j telles que :

$$f_j : X \rightarrow T^j$$

Cette décomposition des classes suppose qu'il existe des regroupements d'exemples qui favorisent la résolution des problèmes.

La notion de sous-classes à proprement parler apparaît dans l'article de Hoffmann et al. (2001) où les auteurs soutiennent la thèse que les performances d'un classifieur sont améliorées si un découpage plus pertinent des données est fourni : *« if the provided examples can be divided into 'meaningful' subclasses then using the subclass label to train a learner may result in improved accuracy for classifying according to the initial set of two or only a few classes. »*

Cette thèse est étayée par l'utilisation d'arbres de décision de type C4.5 appris sur plusieurs jeux de données réelles. Les auteurs fusionnent arbitrairement certaines classes puis ils montrent que cela dégrade les résultats. Si cette façon de faire est critiquable,

nous pensons tout de même que leur interprétation du phénomène est très pertinente. En effet, d'après eux : *« it is easier for C4.5 to learn target function that assign to each class one or a few convex areas in the object space, as opposed to rather complex shaped areas for only a few classes. »*

Les « few convex areas »dont il est question ici, rejoignent notre idée de classes non-homogènes qui se décomposent en sous-classes homogènes.

Donc nous avons vu dans cette partie la formalisation du problème de décomposition de classes. Nous avons aussi mis en évidence le fait que cette décomposition faisait sens si les sous-catégories étaient constituées de groupes compacts. A présent, intéressons-nous aux méthodes qui remplissent ce rôle.

3.2 Méthodes de découpage pour la classification

Même si la construction d'un modèle de classification n'est pas une tâche de caractérisation, certains algorithmes d'apprentissage fonctionnent en découpant l'espace des données en différentes zones. Nous pensons que ces zones peuvent être assimilées à des sous-classes. C'est pourquoi nous allons étudier cette famille de méthodes.

Ainsi, ce qui caractérise ces algorithmes et les rend intéressants à nos yeux c'est qu'ils tracent des *frontières* autour des exemples - sous la forme de sphères ou d'hyperplans.

3.2.1 Classifieurs à base de sphères

Un des premiers classifieurs à base de sphères à avoir été proposé est l'algorithme du *Reduced Coulomb Energy* (RCE) que l'on peut voir dans l'ouvrage de Duda et al. (2000). Son fonctionnement est très intéressant et se rapproche des premières phases de la méthode de découpage que nous proposons dans nos travaux :
On se fixe un rayon maximal r_{max}. Puis, pour chaque exemple X_i de classe C_i du jeu de données, on définie la sphère S_{X_i} centrée en X_i dont le rayon est égal soit à la distance qui sépare X_i de X_j avec X_j l'exemple le plus proche de classe différente - soit égale à r_{max}.
La phase de classification est assez simple : pour chaque nouvel exemple, on regarde dans quelle sphère il se trouve et on lui attribue la classe correspondante.

Ainsi, l'algorithme du RCE propose un pavage de l'espace selon des sphères. Cependant, le nombre de sphères est égal au nombre d'exemples présents dans le jeu de données ce qui peut poser des problèmes de passage à l'échelle. On trouve dans les travaux de

Wang et al. (2006) une étude sur les méthodes qui minimisent le nombre de sphères utilisées pour recouvrir les données.

Le problème de pavage de l'espace par des sphères est NP-complet (Garey & Johnson, 1979). Cependant, un algorithme glouton est connu pour donner de bons résultats. Son fonctionnement est le suivant :
Dans la phase d'initialisation, on applique l'algorithme du RCE pour constituer l'ensemble des sphères $S = \{S(X_1), \ldots, S(X_n)\}$ et on compte combien d'exemples chaque sphère couvre : $N(X_i)$. Puis, on ajoute une à une les sphères dans l'ordre des $N(X_i)$ décroissants jusqu'à couvrir l'ensemble des exemples.

La minimisation du nombre de sphères est maximale dans les travaux de Wang et al. (2005) où un noyau est utilisé pour changer l'espace de représentation et où une unique sphère permet d'obtenir de bons résultats.

La dernière méthode de classifieurs à base de sphères qui représente un grand intérêt dans la décomposition des classes, est l'algorithme des *Set Covering Machine* (SCM) développé par Marchand et Taylor (2003). Les auteurs se focalisent sur les problèmes à deux classes. Pour chaque classe, l'algorithme trouve la (ou les) sphères qui couvrent le mieux les exemples. L'originalité de cette approche est que le recouvrement de sphères est autorisé, voire encouragé, car le modèle final est représenté par l'ensemble des conjonctions et des disjonctions des sphères.

On peut voir une illustration en deux dimensions de la méthode sur la figure 3.2 qui est un extrait de transparents réalisés par Dale et James (2003). On cherche à apprendre à reconnaître les exemples positifs. La sphère $b1$ couvre les exemples positifs de l'échantillon d'apprentissage. Les sphères $b2$ et $b3$ couvrent les exemples négatifs qui sont dans la sphère $b1$. Ainsi le modèle de reconnaissance f est représenté par la sphère $b1$ privée de $b2$ et $b3$.

De plus, les auteurs des SCM fournissent une solide étude sur l'erreur de généralisation de leur algorithme dans (Marchand & Shawe-Taylor, 2001) et (Marchand & Taylor, 2003) ainsi qu'une comparaison de leurs résultats avec les Support Vector Machine.

3.2.2 Arbres de décision

En marge des méthodes de classification à base de sphères on trouve des algorithmes qui découpent l'espace des données par le biais d'hyperplans : ce sont les méthodes à base d'arbres de décision.

Les arbres de décision ont été formalisés par Breiman et al. (1984) pour le cas des arbres binaires puis enrichis par Quinlan (1993) pour construire des arbres de décision

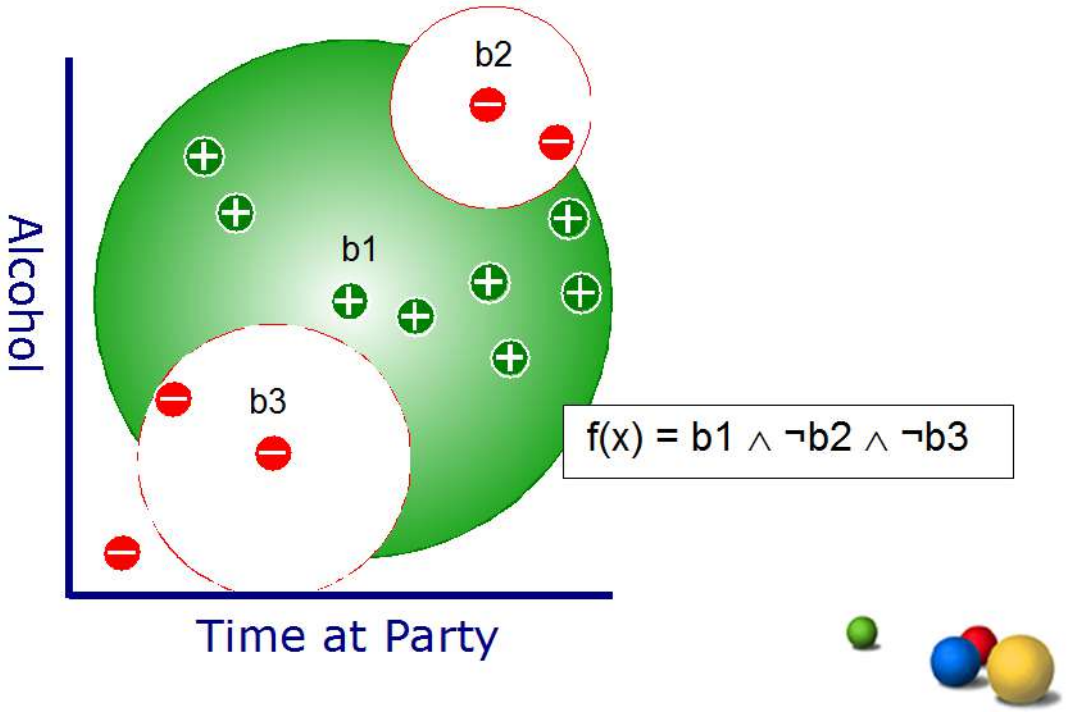

FIGURE 3.2 – Extrait de transparents de présentation des SCM. Le classifieur des exemples positifs est donné par la fonction f.

« quelconques. » Le principe de l'algorithme est d'utiliser un critère d'entropie pour choisir l'attribut le plus discriminant et tracer des hyperplans séparateurs représentés par des nœuds de l'arbre.

L'intéressant avec ce type de classifieurs est que la prise en compte des classes non homogènes est implicite. En effet, aucune supposition de compacité des groupes n'est requise. Ainsi, l'utilisation des arbres comme méthode de décomposition est possible : il suffit de voir que chaque feuille de l'arbre est une zone de l'espace dédiée à quasi exclusivement une classe. C'est donc un outil qui a sa place dans les méthodes de segmentation, même si nous n'avons pas rencontré dans la littérature une utilisation des arbres pour la caractérisation de classes.

3.3 Méthode de découpage explicites

Nous venons de voir les méthodes de classification qui peuvent être utilisées, en les modifiant légèrement, pour décomposer l'espace des données et donc découvrir les sous-classes des classes non homogènes.

Nous allons à présent nous focaliser sur les méthodes dont le but est explicitement de fournir un découpage des classes en sous-classes. Le point commun de toutes ces approches est qu'elles utilisent un algorithme de classification non supervisé (*clustering*) pour découper les classes.

L'idée de l'utilisation d'une méthode de clustering pour trouver les sous-classes est énoncée par Japkowicz (2002). Son objectif est de segmenter un jeu de données puis de faire plusieurs modèles sur ce nouveau jeu et de combiner les décisions des classifieurs. En pratique l'auteur utilise les k-moyennes (Macqueen, 1967) pour segmenter chaque classe séparément.

Une approche similaire se retrouve dans les travaux de Vilalta et Rish (2003) et de Vilalta et al. (2003) où le clustering est utilisé pour augmenter les performances de classifieurs, notamment ceux de type *Naive Bayes*. Les auteurs expliquent les gains obtenus de la façon suivante : *« Decomposing classes into clusters makes the new class distribution easier to approximate and provides a viable way to reduce bias while limiting the growth in variance. »*

Une autre série de travaux très intéressants ont été réalisés par Eick et Zeidat (2005) et Eick et al. (2006). Les auteurs utilisent ce qu'ils nomment du « clustering supervisé » [1] pour faire du résumé d'information. Dans leurs travaux, le clustering est réalisé sur la globalité des données. Pour chaque cluster identifié, un prototype est choisi pour le représenter.

Ce processus est illustré sur la figure 3.3 empruntée à leur article, où on voit le cheminement de la décomposition puis du résumé d'un jeu de données en deux dimensions.

La finalité est une fois encore d'aider un classifieur de type Naive Bayes, mais l'action de résumer des classes par des prototypes s'inscrit clairement dans un processus de caractérisation de classes et de découverte de sous-classes pour y parvenir.

Enfin, nous terminons notre état de l'art des méthodes explicites de construction de sous-classes en traitant des travaux de Wu et al. (2007) dans lesquelles la détection de sous-classes par clustering sur chaque classe est utilisée dans le but de pallier le problème des classes non équilibrées. En effet, en présence de classes de tailles très différentes, les algorithmes d'apprentissage peuvent ne pas fonctionner correctement et être biaisés. Dans

1. Cette appellation est non appropriée car le clustering supervisé est un domaine existant (Basu et al., 2004) où un petit nombre d'exemples étiquetés va guider le regroupement d'exemples non étiquetés - ce qui n'est pas le cas ici.

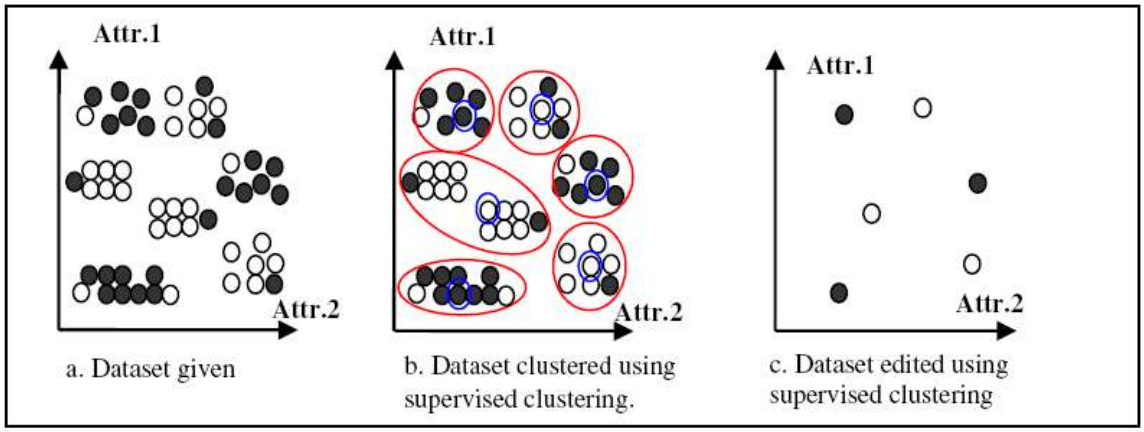

FIGURE 3.3 – Illustration du processus de résumé par prototypes de sous-classes dans les travaux de Eick et al. (Figure empruntée de leur article.)

l'article qui nous intéresse, il est suggéré que plutôt d'utiliser des méthodes de type *bagging* (Breiman, 1996) qui utilisent plusieurs fois certains exemples du jeu de données, il peut être intéressant de découper le jeu de données en sous-classes de taille homogène pour gommer la différence de taille.

3.4 Synthèse

Comme nous l'avons vu dans ce chapitre il existe deux points de vue très distincts dans le domaine de la décomposition de classes.

Le premier s'attache à *tracer des frontières à l'aide de sphères ou d'hyperplans* dans l'espace des données pour délimiter des zones et les étiqueter. Le but est de fournir un modèle de généralisation pour faire de la classification. Notre principale critique de ce type de méthodes est la suivante : ces approches ne se placent pas dans un cadre de caractérisation de données et elles ne sont pas dédiées à cette tâche. Par exemple, elles ne fournissent pas d'affectation à des sous-classes. La caractérisation et la généralisation ne sont, de plus, pas toujours compatibles.

Le second point de vue est l'utilisation d'une *méthode de clustering sur chaque classe séparément ou sur la globalité des données*, pour effectuer un prétraitement des données. Le nouveau découpage est alors utilisé pour améliorer les performances des classifieurs. Malgré un principe intuitif, nous voyons deux critiques sérieuses :

– Les méthodes de clustering requièrent généralement qu'on leur donne en paramètre le nombre de groupes recherchés. Cela suppose donc que l'on a déjà identifié et caractérisé les classes non homogènes, ce qui est un non sens.

– De plus, soit le clustering est effectué sur chaque classe indépendamment (Japkowicz, 2002; Vilalta & Rish, 2003; Vilalta et al., 2003) et dans ce cas l'information de dynamique des classes les unes par rapport aux autres et notamment celle sur les zones de chevauchement est complètement perdue ; soit le clustering est calculé globalement (Eick & Zeidat, 2005) ce qui revient à perdre l'information de classe.

On s'aperçoit donc que les deux approches ont des avantages et des inconvénients. Dans la suite du document nous allons présenter notre méthode de décomposition qui s'inscrit plutôt dans la première logique tout en gardant le coté explicite de la seconde.

Deuxième partie

Notre proposition pour la découverte de sous-classes

Chapitre 1

Notre approche

1.1 Introduction

Dans ce chapitre nous proposons un algorithme original pour enrichir la caractérisation des classes en détectant de façon automatique les sous-classes.

Comme nous l'avons vu précédemment, une catégorie d'objets peut rassembler des exemples ayant différents comportements ; c'est-à-dire que, bien qu'ayant des descriptions très différentes, les exemples appartiennent à la même classe.

Nous pouvons rappeler notre exemple de la catégorie des bons élèves vu dans la partie 2.1 page 30 : Il y a la sous-classe des élèves doués et la sous-classe de ceux qui travaillent dur. La catégorie est la même, mais les descripteurs mis en œuvre (i.e. les comportements) sont très différents. La classe des bons élèves est donc une classe non-homogène composée de deux sous-classes (c'est-à-dire deux groupes d'exemples). Nous partons de l'hypothèse que les groupes d'exemples des classes non homogènes sont éloignés les uns des autres et qu'il y a donc des chances que des exemples d'autres classes les *séparent*. Cette notion de séparation est le pivot sur lequel s'appuie notre méthode, c'est sur cette propriété que nous nous appuyons pour construire les sous-classes. A l'instar de la typicalité nous nous servons des autres classes pour proposer un découpage en sous-classes.

De plus, nous défendons l'idée qu'une sous-classe n'est pas uniquement un groupe d'exemples d'une même classe, mais bel et bien *une zone de l'espace des données où une classe est très majoritaire.*

Dans le but de garantir une tolérance au bruit et permettre à l'utilisateur un réglage de la méthode, nous avons introduit le paramètre de tolérance t.

Dans la suite de ce chapitre, après avoir brièvement décrit le contexte d'application, nous exposerons en détail, notre approche. Dans un second temps, nous illustrerons les propriétés de celle-ci sur des données artificielles avant de conclure.

1.2 Notations et prérequis

L'utilisation de notre approche nécessite une base de données numériques. Ainsi, nous supposons que nous travaillons sur une base de données D contenant n exemples :

$$D = \{X_1, X_2, ..., X_n\}$$

Chaque exemple X_i est décrit par m attributs numériques [2] :

$$X_i = \{x_{i,1}, x_{i,2}, ..., x_{i,m}\}, x_{i,j} \in \mathbb{R}$$

De plus, nous nous plaçons dans un cadre supervisé. Il existe donc p classes : C_k - et chaque objet X_i appartient à l'une d'entre elles :

$$\forall X_{i=1...n}, \exists k \in \{1, \ldots, p\} | X_i \in C_k$$

$$et \bigcup_{k=1}^{p} C_k = D$$

Enfin, nous supposons que nous avons à notre disposition une mesure de distance d.

1.3 Etape préliminaire : normalisation des données

Dans le but de ne pas avantager un attribut par rapport à un autre, nous avons choisi de normaliser nos données. Pour cela nous utilisons une des normalisations qui consiste à centrer et réduire nos données avant de les traiter. Pour cela nous calculons, pour chaque attribut A, sa moyenne m_A et son écart type ρ_A.

Puis, nous définissons pour chaque valeur V de l'attribut A sa nouvelle valeur V' :

$$V' = \frac{V - m_A}{\rho_A}$$

A la suite de cette opération notre jeu de données a une moyenne nulle et un écart type qui vaut 1. De ce fait, les distances entre exemples sont indépendantes de l'échelle des différents attributs, ce qui garantit une comparaison fiable entre attributs.

Enfin, la dernière étape préliminaire consiste à calculer la matrice M des distances [3] entre tous les exemples.

$$M_{i,j} = d(X_i, X_j)$$

Notons que notre algorithme travaille exclusivement sur la matrice des distances. Il est donc possible de se passer de la base de données une fois que M est disponible.

2. Il est également possible de prendre en compte les attributs binaires.
3. En pratique nous ne calculons que la demi-matrice car celle-ci est symétrique.

1.4 Première étape : zones d'influence

La première étape de notre approche consiste à définir, pour chaque exemple X_i, sa *zone d'influence*. Etant donné le paramètre de tolérance réglé à la valeur t, la zone d'influence de X_i est définie comme l'hypersphère centrée en X_i, de rayon maximal, qui contient t exemples qui ne sont pas de la classe de X_i.

En d'autres termes, cette zone est l'hypersphère dont le rayon est égal à la distance qui sépare X_i du $t+1$ ième plus proche exemple d'une autre classe que la sienne.

La figure 1.1 illustre cette étape de notre méthode sur une base d'exemples en deux dimensions séparés en deux classes. La zone violette est la zone d'influence de l'exemple central. Suivant le réglage du paramètre de tolérance, on trouve t croix rouges dans la zone. Trois valeurs de t sont représentées sur la figure : $t = 0$, 1 et 2. La zone d'influence augmente avec le paramètre de tolérance.

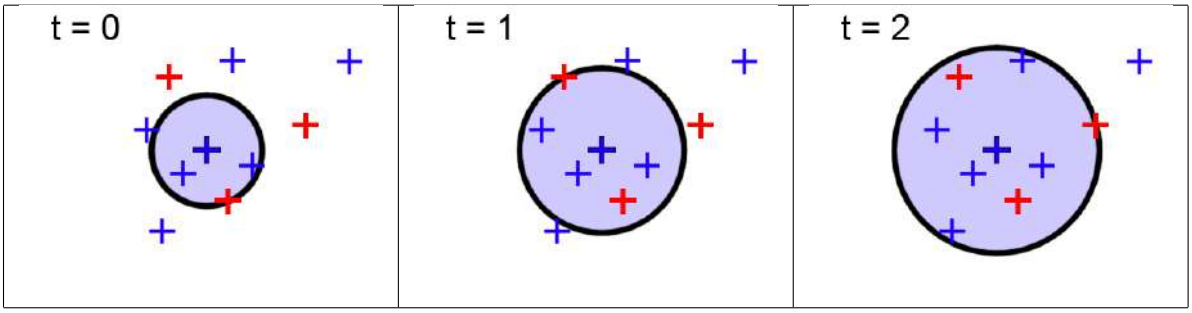

FIGURE 1.1 – Illustration des sphères d'inflence selon trois réglages de la tolérance t.

1.5 Deuxième étape : création des graphes d'amis directs

Grâce à l'étape précédente, nous disposons, pour chaque exemple X_i de la base de données, de sa zone d'influence. Nous appelons *amis directs de* X_i, les exemples de la même classe que X_i qui sont présents dans la zone d'influence. Ce sont des exemples proches qui ne sont pas (ou peu) séparés de X_i.

La seconde étape de notre méthode est la création, pour chaque exemple, de liens avec ses amis directs. Pour représenter ces liens, travailler dessus et les garder en mémoire, nous utilisons le formalisme des graphes. Ainsi, nous définissons G_{X_i} le graphe des liens de l'exemple X_i : nous débutons par la création d'un nœud pour l'exemple X_i. Puis, pour

tout exemple X_j présent dans la zone d'influence de X_i, si les deux exemples sont de la même classe, nous créons un nœud pour X_j ainsi qu'une arête non orientée $A : X_i \leftrightarrow X_j$.

Une illustration de ce processus se trouve sur la figure 1.2. On y voit la construction du réseau d'amis directs d'un exemple pour une valeur de tolérance égale à 2. Le graphe correspondant contient six nœuds et cinq arêtes.

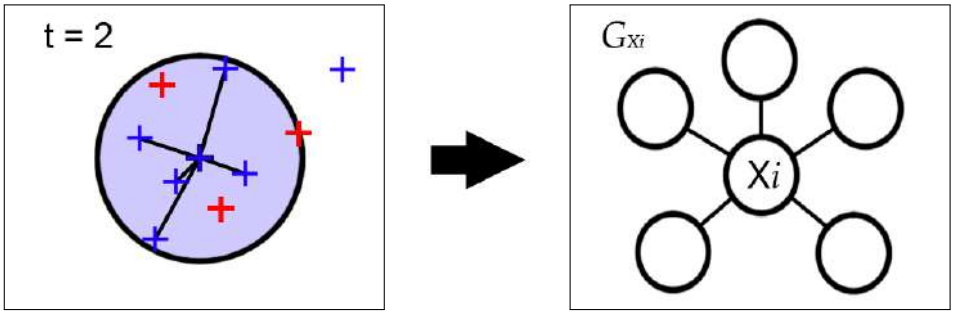

FIGURE 1.2 – Illustration de la création du graphe d'amis directs de l'exemple X_i.

1.6 Troisième étape : union des graphes d'amis directs

En utilisant conjointement tous les graphes, on obtient un graphe global qui permet de relier entre eux les exemples de proche en proche. Dans ce but, nous définissons G, le graphe global de liens d'amitié comme l'union de tous les graphes d'amis directs :

$$G = \bigcup_{i=1}^{n} G_{X_i}$$

La figure 1.3 illustre l'union de trois graphes. On constate que des chemins se créent entre les exemples éloignés grâce aux liens de proche en proche. L'information qui résulte de cette union est extrêmement riche. En effet, en assemblant tous les graphes individuels, on assiste à l'émergence de communautés d'exemples - prémisses des sous-classes que nous recherchons. Nous appelons ces communautés des *réseaux d'amis*.

Nous définissons l'« amitié » comme une relation d'équivalence $\star$: deux exemples X_i et X_j sont amis $(X_i \star X_j)$ si *l'un appartient à la zone d'influence de l'autre* ou si *l'un appartient à la zone d'influence d'un des amis de l'autre.*

Comme toute relation d'équivalence, trois propriétés s'appliquent :

- Refléxivité : chaque exemple est son propre ami. $X_i \star X_i$
- Symétrie : l'amitié n'est pas à sens unique. $X_i \star X_j \Leftrightarrow X_j \star X_i$
- Transitivité : les amis de mes amis, sont mes amis. $X_i \star X_j$ et $X_j \star X_k \Rightarrow X_i \star X_k$

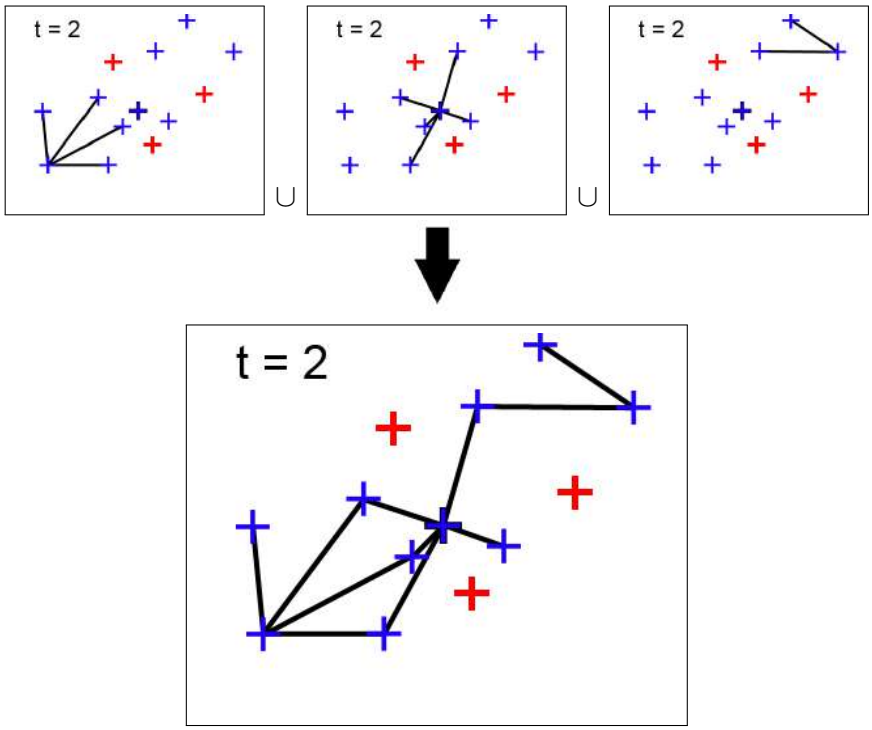

FIGURE 1.3 – Union de trois graphes d'amis directs.

1.7 Quatrième étape : affectation aux sous-classes

La quatrième et dernière étape de notre méthode est l'assignation des exemples à des sous-classes. Pour cela, nous nous appuyons sur le graphe de relations d'amitié. En effet, nous définissons les sous-classes comme *l'ensemble des graphes connexes au graphe d'amis*. Cela signifie que si deux individus sont amis, alors ils appartiennent à la même sous-classe - ou encore, que s'il existe un chemin dans le graphe d'amis qui relie deux nœuds, alors les exemples correspondant sont dans la même sous-classe.

C'est en utilisant un algorithme de parcours de graphe que nous assignons à chaque nœud, et donc à chaque exemple, la sous-classe qui le contient. Nous procédons de façon itérative en prenant les exemples dans l'ordre du jeu de données. Cependant, ce choix est purement arbitraire car l'ordre n'influe pas sur les résultats d'affectation.

1.8 Complexité de la méthode proposée

Notre algorithme a une complexité en $O(n^2 Log(n))$ où n est le nombre d'exemples présents dans le jeu de données. En effet, si nous décomposons chaque étape :

- La première étape est le calcul de la matrice de distances. Elle nécessite de comparer tous les exemples deux à deux et a donc une complexité en $O(n^2)$.
- Puis, la construction des zones d'influences requiert, pour les n exemples, de trier les vecteurs de distances par ordre croissant ce qui, au total, s'effectue avec une complexité en $O(n)O(nLog(n)) = O(n^2Log(n))$.
- Enfin, le parcours du graphe d'amis nécessite d'emprunter chaque arête ce qui prend un temps constant en fonction du nombre d'arêtes : $O(|F|)$ où $|F|$ est le nombre de relations d'amitié.

Il est possible d'améliorer le temps de traitement de notre méthode en parallélisant l'algorithme sur plusieurs machines. En effet, le traitement de chaque classe peut être effectué indépendamment. On obtient alors une complexité en $O(cnLog(n))$ où n est le nombre d'exemples total et c le nombre d'exemples dans la classe qui est traitée. Dans le cas de classes équilibrées, cela revient à diviser le temps de calcul par le nombre de classes, ce qui n'est pas négligeable.

En outre, la parallélisation de l'algorithme permet de diminuer l'occupation mémoire de la matrice de distances et la liste d'arêtes du graphe qui, dans le cas de grands jeux de données, peuvent s'avérer trop volumineuses.

Chapitre 2

Illustrations de notre approche

Dans ce chapitre, nous détaillons les principales caractéristiques de notre approche. Notre but est de mettre en avant ses spécificités ainsi que les comportements originaux qui la font se différencier des méthodes existantes.

Pour ce faire, nous utilisons des jeux de données artificielles généralement découpés en deux classes. Dans chaque illustration, nous représentons tout d'abord le jeu de données, puis nous dessinons le graphe des relations d'amitié pour enfin finir avec l'affectation des sous-classes. Pour chacune de ces illustrations, à moins que nous précisions le contraire, le paramètre de tolérance est réglé à 0.

Dans un souci de clarté, nous avons choisi de représenter le graphe des relations d'amitié sous une forme simplifiée grâce à l'algorithme des arbres couvrants minimum (Cormen et al., 1994). Les résultats restent inchangés alors que la lisibilité est grandement accrue.

Notre première illustration consiste à montrer que notre approche détecte efficacement les différents comportements des classes non homogènes. Puis, dans un second temps, nous nous intéressons au cas des classes compactes : celles-ci ne nécessitent pas de segmentation et notre méthode respecte cette contrainte. La troisième illustration se focalise sur la gestion efficace du bruit alors que la quatrième se concentre sur la forme des groupes d'exemples. Enfin, le dernier comportement que nous souhaitons mettre en avant est la gestion des zones de chevauchement entre classes.

2.1 Découverte efficace des comportements

Le but de cette section est de montrer que notre approche détecte de façon satisfaisante les différents comportements des classes non homogènes. Pour illustrer cette propriété essentielle, nous utilisons un jeu de données artificielles qui comporte une classe compacte au centre et une classe non homogène répartie en deux sous-groupes de part et d'autre.

La figure 2.1 représente le jeu de données. Celui-ci est composé de 200 exemples - 100 de chaque classe.

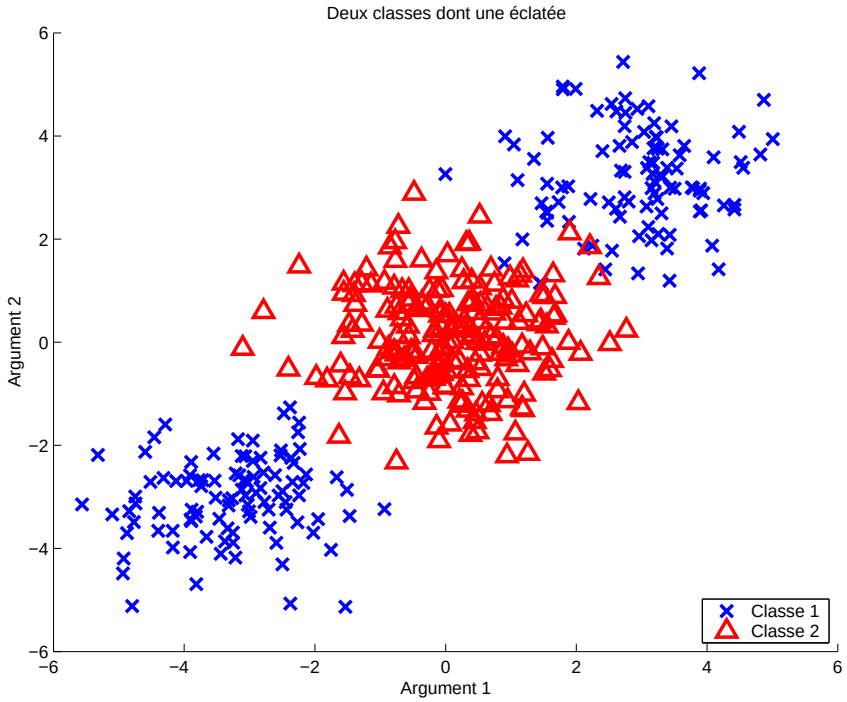

FIGURE 2.1 – Jeu de données artificiel : Deux classes sont représentées dont une non-homogène.

En appliquant notre approche sur ce jeu de données nous obtenons le graphe de la figure 2.2 en haut. On constate que le graphe est composé de trois sous-graphes connexes. Les différents comportements de la classe non homogène sont donc correctement restitués.

En parcourant le graphe et en attribuant les sous-classes, nous obtenons l'affectation visible sur la figure 2.2 en bas. 8 sous-classes sont découvertes dont 3 qui contiennent la quasi totalité du jeu de données. Les cinq autres correspondent aux zones de chevauchement des classes.

Notre approche atteint donc l'objectif pour lequel nous l'avons conçue : découvrir efficacement les sous-groupes des jeux de données. Les zones de l'espace qui « appartiennent » de façon presque exclusive à une classe sont bien identifiées.

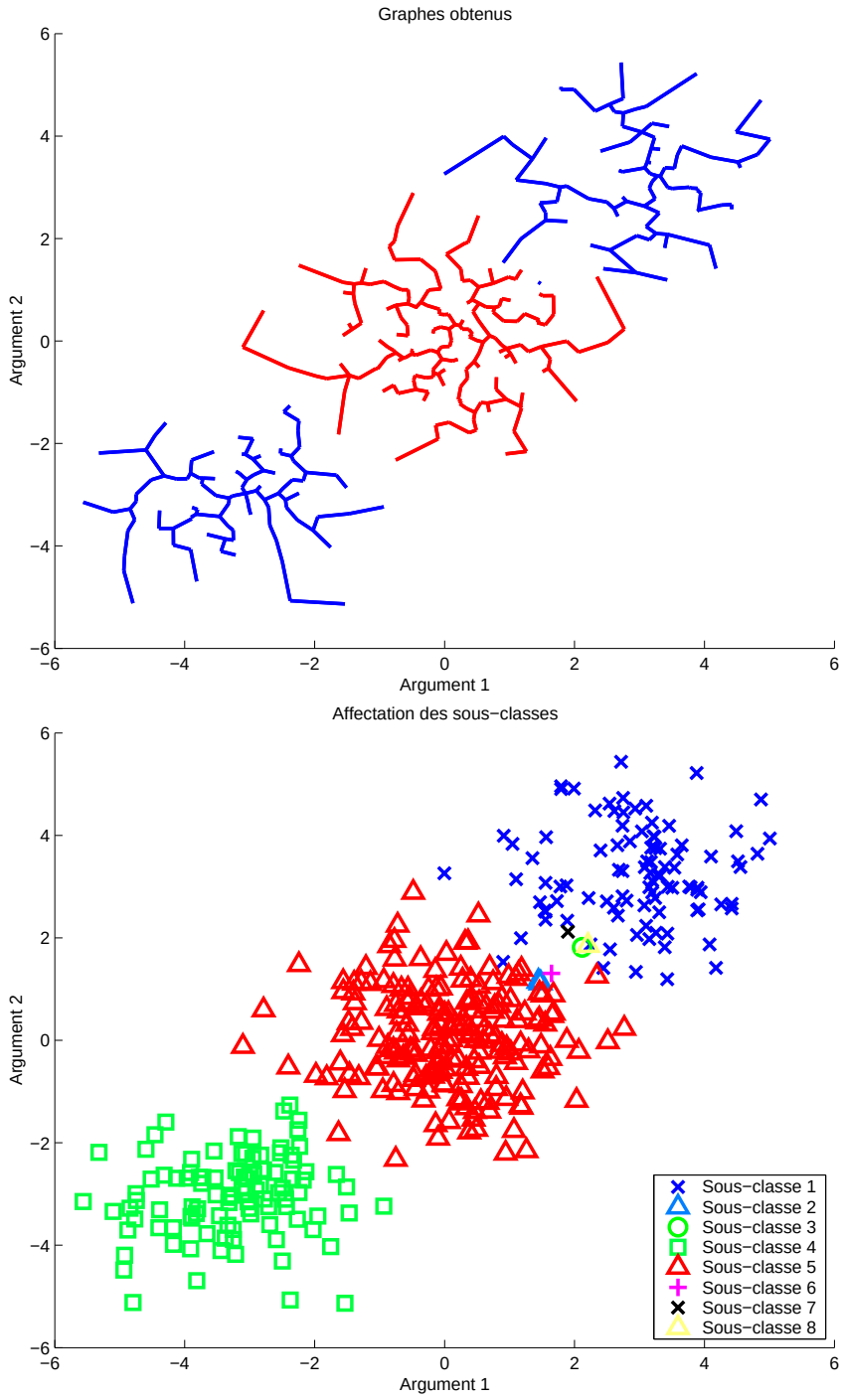

FIGURE 2.2 – Graphe des liens d'amitié entre exemples et sous-classes obtenues.

2.2 Trois groupes distincts

Si notre approche permet une segmentation des classes, il faut prendre garde à ce que cette opération ne se produise pas sur des classes qui n'en ont pas besoin. Sur la figure 2.3 nous pouvons voir la représentation d'un jeu de données artificielles composé de trois classes. Aucune n'est non-homogène - il n'y a donc pas besoin d'effectuer une segmentation.

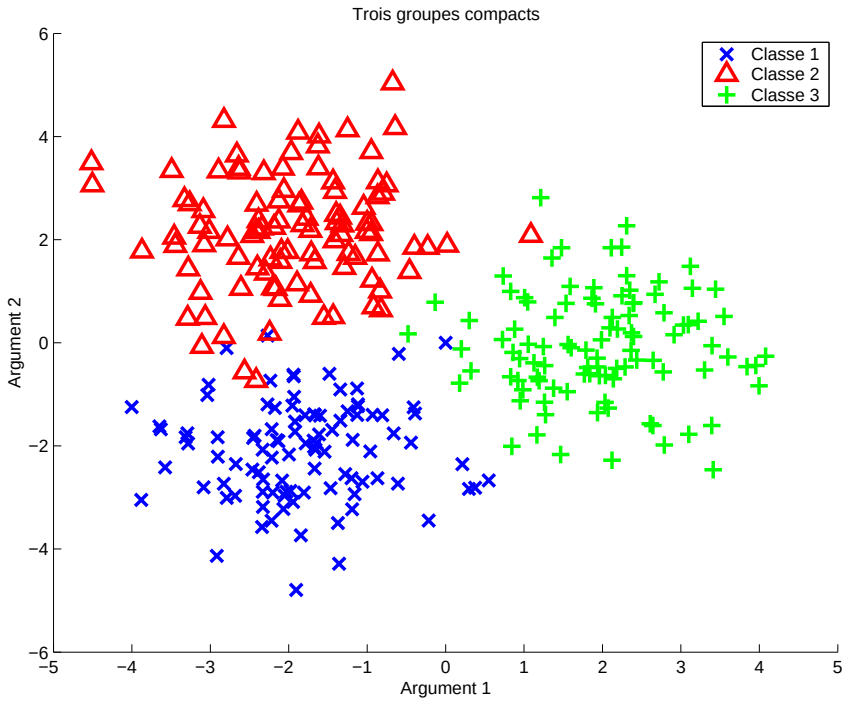

FIGURE 2.3 – Jeu de données artificielles : trois classes homogènes

La figure 2.4, en haut, représente le graphe d'amitié que notre approche renvoie sur les trois classes compactes. On constate qu'il est constitué de quatre sous-graphes connexes dont un n'est qu'une arête.

En faisant l'affectation des sous-classes (figure 2.4 en bas) on a la confirmation que les trois classes n'ont presque pas été divisées par notre algorithme. En effet, mis à part 5 sous-classes composées d'un unique exemple et une autre rassemblant deux exemples, trois sous-catégories sont mises en avant.

Ainsi, ce jeu de données illustre ce que nous appelons le principe de parcimonie : notre méthode n'effectue pas de découpage lorsqu'il n'y en a pas besoin. En effet, une segmentation non souhaitée dégraderait la caractérisation.

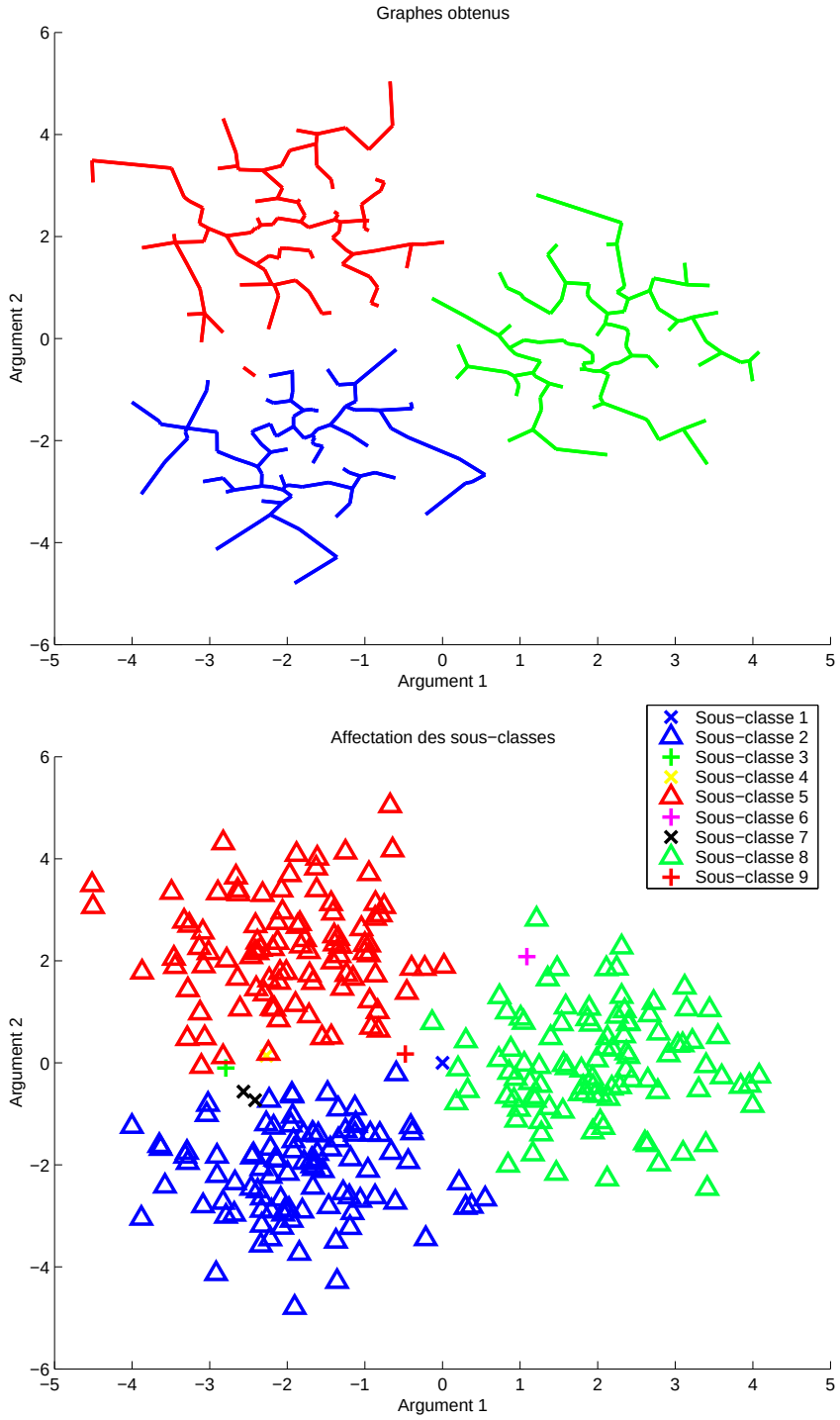

FIGURE 2.4 – Graphe des liens d'amitié entre exemples.

2.3 Gestion du bruit

Un des avantages majeurs de notre approche est sa robustesse. Cela est rendu possible grâce à la procédure de construction des zones d'influence ainsi qu'à l'union des graphes d'amis directs. En se regroupant de proche en proche, les exemples « contournent » l'éventuel bruit qu'il peut y avoir dans les données.

La figure 2.5 montre un jeu de données artificielles constitué d'une classe homogène parsemée de bruit. La classe principale est composée de 100 exemples alors que la classe parasite compte 10 exemples.

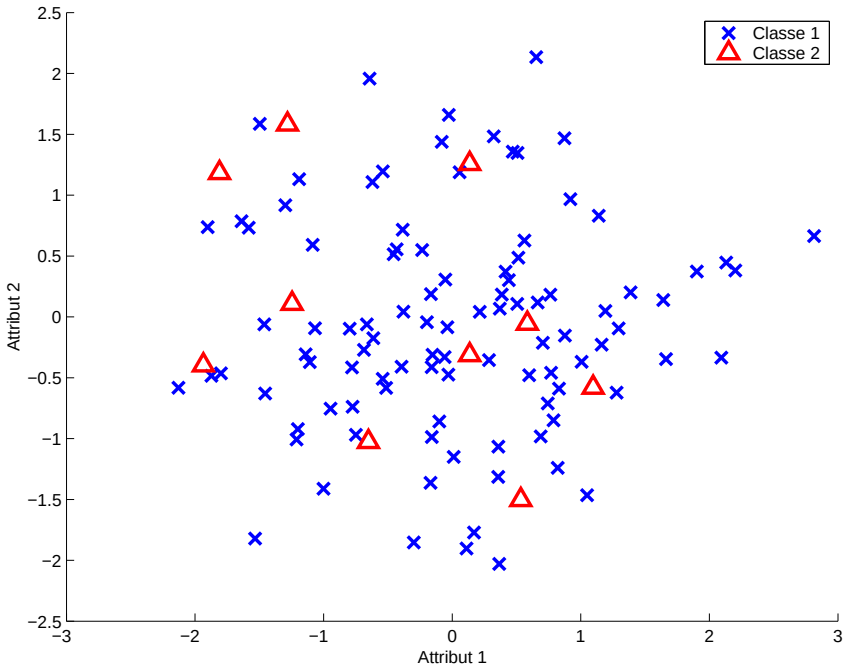

FIGURE 2.5 – Une classe homogène bruitée par des exemples d'une autre classe

En appliquant notre méthode sur ce jeu de données on constate que les exemples qui appartiennent à la classe parasite gênent la progression des zones d'influence. Ainsi, des liens d'amitié qui devraient normalement exister sont absents. Mais puisque deux exemples sont de la même sous-classe s'il existe au moins un chemin qui les relie, ces liens directs manquants sont en grande majorité remplacés par des chemins plus longs mais tout aussi valides.

Sur la figure 2.6, en haut, nous avons dessiné tous les liens d'amitié (et non plus uniquement l'arbre couvrant comme précédemment). On constate des « trous » là où il y a du bruit. Cependant, à deux exceptions près, les exemples sont tout de même reliés les uns aux autres en contournant le bruit.

La figure 2.6, en bas, montre les sous-classes obtenues. La classe homogène n'est amputée que de 2 exemples sur 100 - ce qui est très satisfaisant étant donné le degré de bruitage subi. Ces exemples ont été séparés car ils sont en périphérie de la classe. Aucun chemin de substitution n'est disponible pour les rattacher au reste de la classe.

La solution pour prendre en compte une plus grande quantité de bruit consiste à augmenter le paramètre de tolérance. En effet, pour une tolérance égale à 1, la classe garde toute son intégrité.

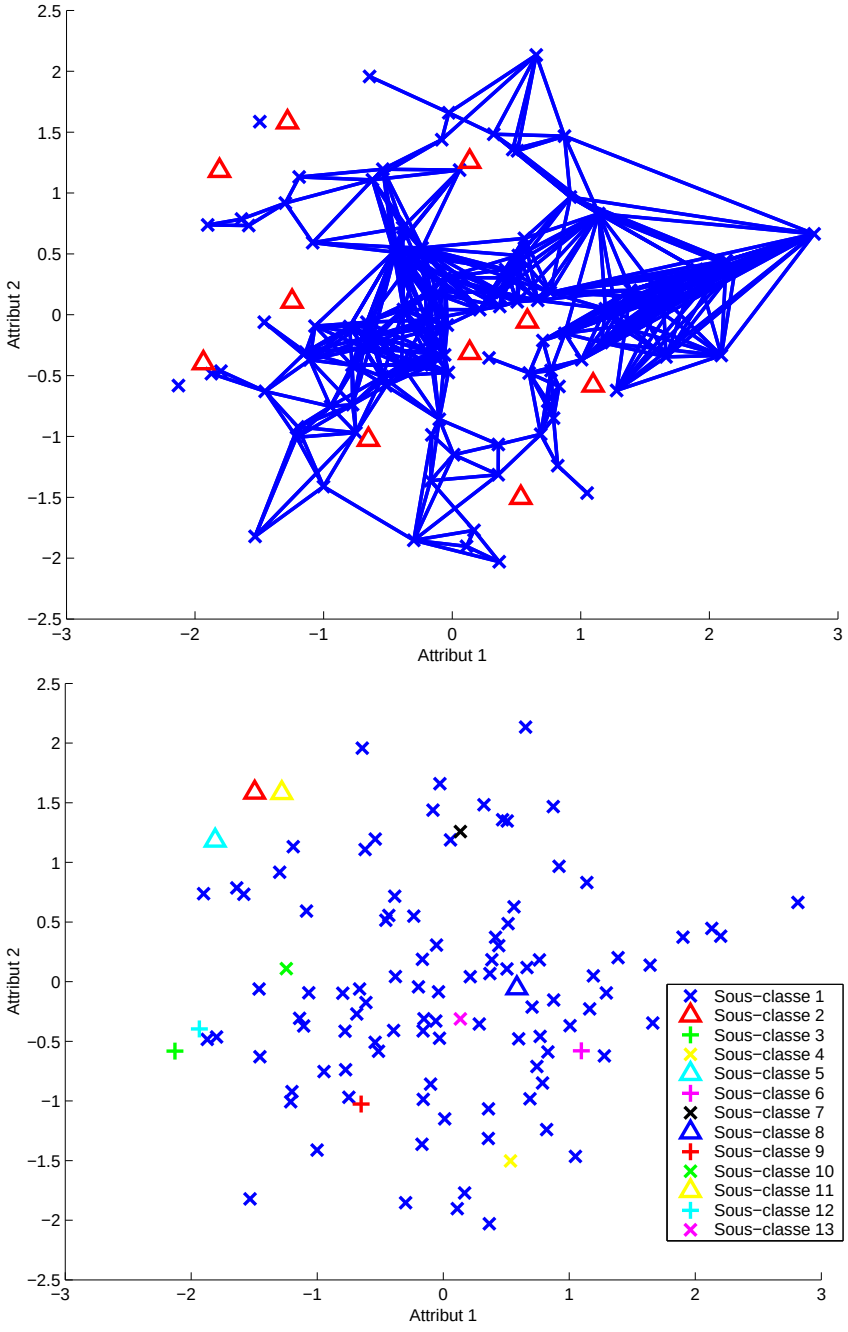

FIGURE 2.6 – Graphe des liens d'amitié entre exemples.

2.4 Forme des groupes

Contrairement à des méthodes qui travaillent attribut par attribut comme les arbres de décision, par exemple, notre approche tisse des liens de proche en proche. Ces liens sont créés dans n'importe quelle direction. Cela nous permet d'obtenir des sous-classes même si les données ne forment pas un groupe sphérique. C'est un avantage certain car, en général, rien ne garantit que les données traitées soient constituées de groupes sphériques.

La figure 2.7 représente un jeu de données artificielles constitué de deux classes dont les exemples forment deux spirales enroulées l'une sur l'autre. Une méthode de segmentation opérant uniquement sur des groupes sphériques (ou même compactes) se mettra à sur-segmenter ce jeu de données, dégradant de ce fait la caractérisation et rendant inutile, voire nuisible, l'information d'affectation des sous-classes.

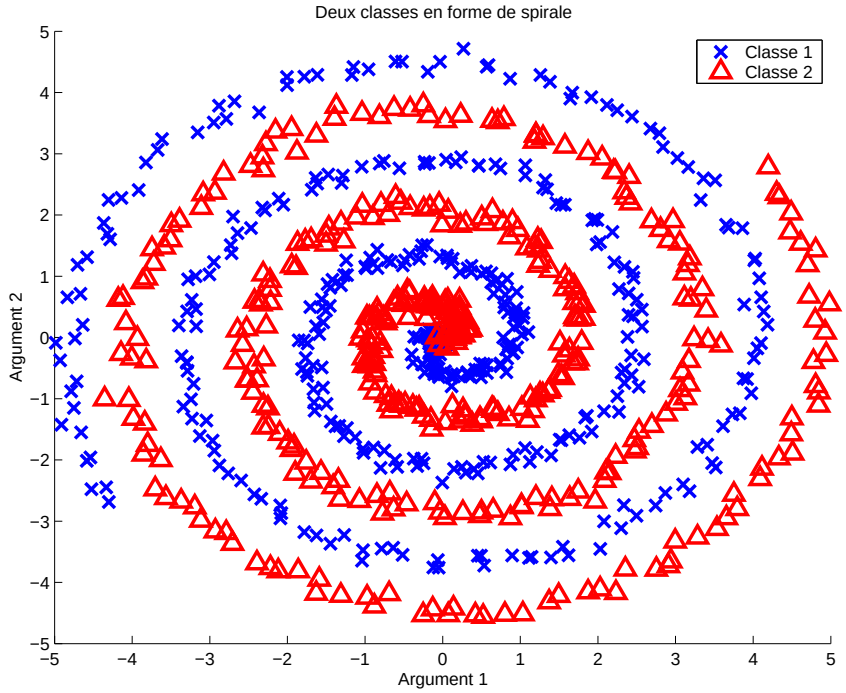

FIGURE 2.7 – Jeu de données artificielles réparties en deux classes et formant deux spirales.

Etant donné que notre méthode procède de proche en proche, nous ne sommes pas limités à la découverte de sous-groupes sphériques. On le constate sur la figure 2.8. Les graphes connexes que nous obtenons sont très allongés mais peu nombreux. Ceci entraîne que les classes ne sont presque pas segmentées et gardent leur intégrité.

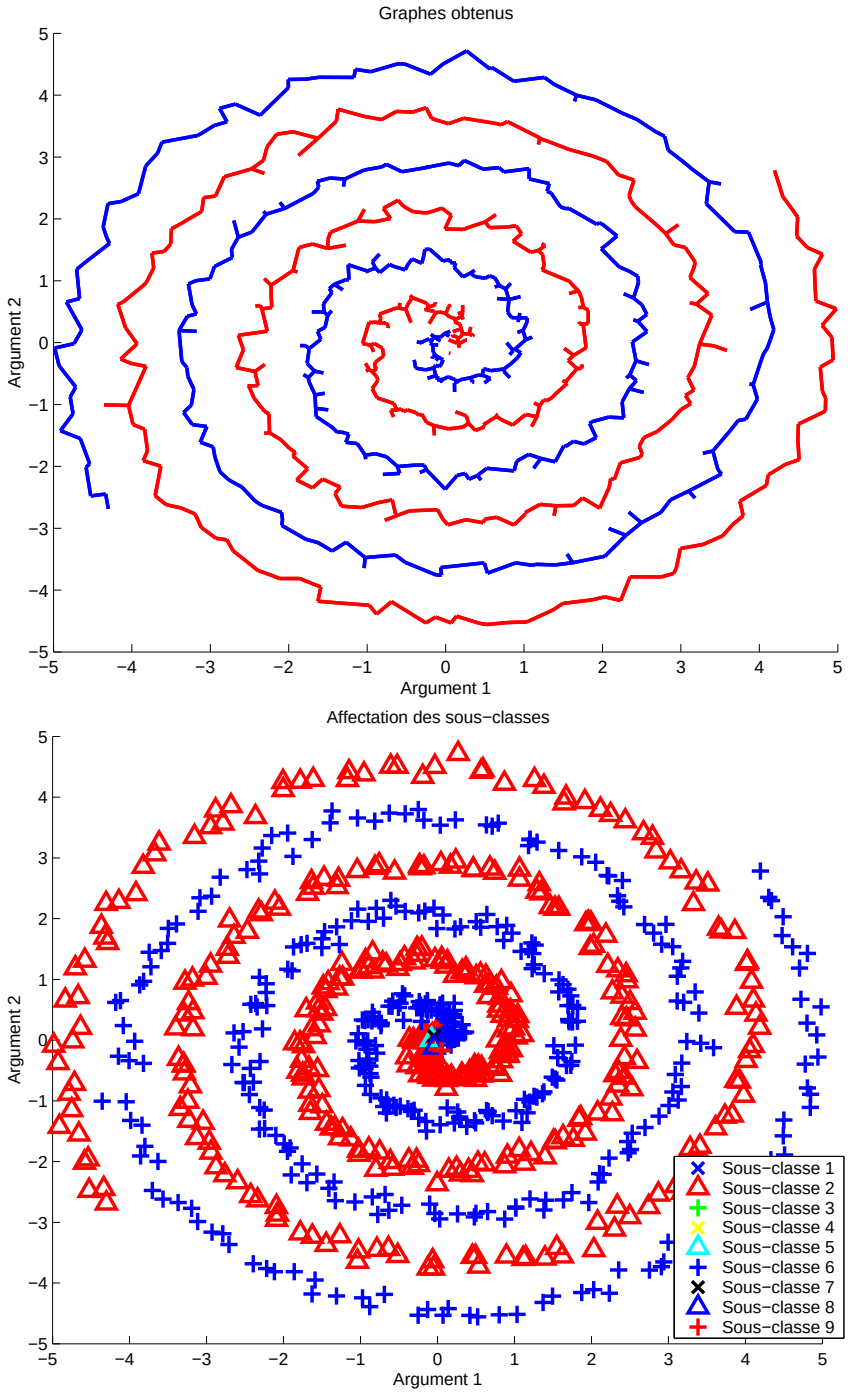

FIGURE 2.8 – Graphe des liens d'amitié entre exemples.

2.5 Construction de sous-classes chevauchantes

Le dernier aspect de notre méthode que nous souhaitons mettre en avant concerne les zones de chevauchement des classes. Comme nous l'avons énoncé précédemment, nous pensons que pour obtenir une bonne caractérisation en sous-classes, il est important de mettre en avant les zones de l'espace des données qui ne contiennent quasiment qu'une seule classe.

Les endroits où les classes cohabitent sont par nature des zones contestées. Les classifieurs ont des difficultés à étiqueter les exemples se trouvant dans ces espaces. Nous pensons qu'il est important de retranscrire dans le découpage en sous-classes cette ambigüité - la caractérisation qui en découlera n'en sera que plus riche de sens.

Un exemple artificiel de ce genre de situation est représenté sur la figure 2.9. Deux classes se croisent au centre du graphique.

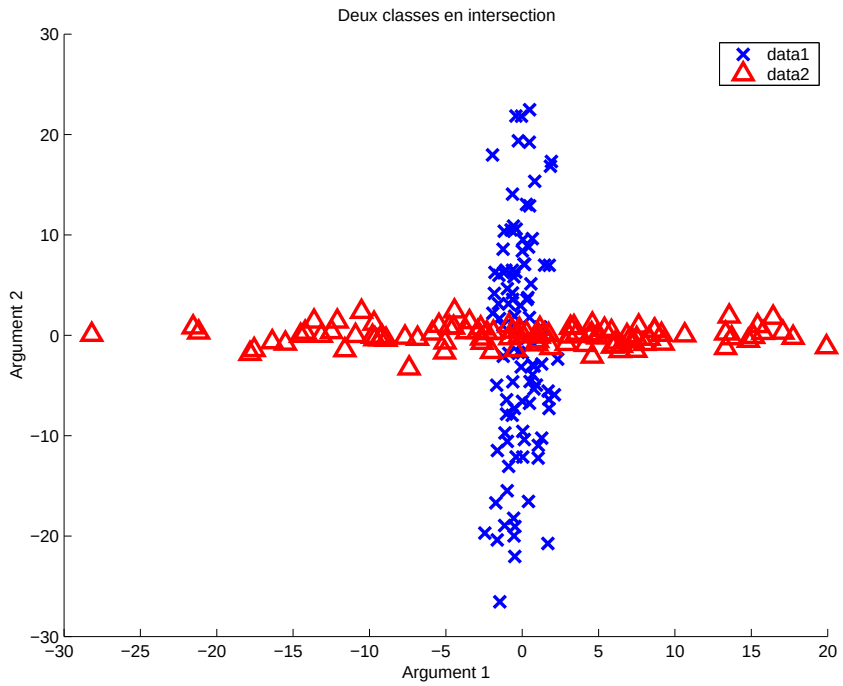

FIGURE 2.9 – Jeu de données artificielles : deux classes qui se croisent

La figure 2.10 montre les résultats après application de l'algorithme sur ce jeu de données : les deux classes ont été découpées en quatre sous-classes de taille relativement élevée et en une multitude de sous-classes ne contenant qu'un seul exemple. Comme nous le souhaitions, la zone contestée est sur-segmentée. De ce fait, on met bien en évidence l'ambigüité que cette intersection contient.

Intuitivement, le fait d'avoir un grand nombre de très petites sous-classes peut laisser

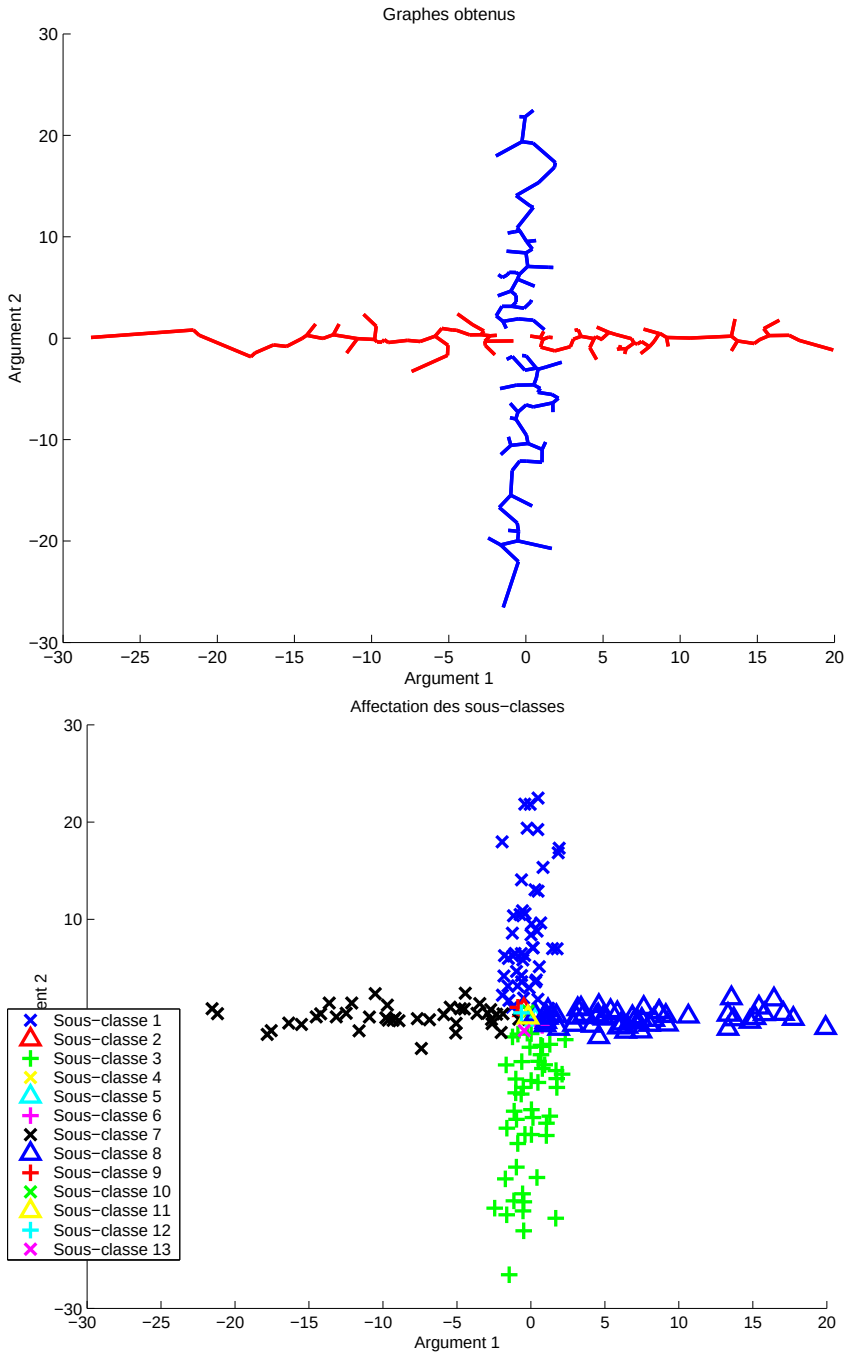

FIGURE 2.10 – Graphe des liens d'amitié entre exemples.

penser que la méthode ne donne pas d'information. Or, c'est bien l'inverse. Notre approche se sert de l'antagonisme entre les classes pour créer les séparations. Un grand nombre de sous-classes signifie donc un degré de mélange élevé - c'est une information très importante pour une tâche de caractérisation ou même d'apprentissage. Cela donne une estimation de la difficulté à étiqueter une zone de l'espace et donc un indice de fiabilité de l'étiquetage.

2.6 Conclusions

Dans ce chapitre, nous avons illustré les points clés de notre travail. Nous avons montré que notre approche est conçue pour détecter efficacement les comportements des classes non homogènes. Nous avons, en outre, mentionné le fait que notre méthode respectait un principe de parcimonie en ne segmentant pas les classes homogènes.

Nous avons également mis en avant la robustesse de notre algorithme en illustrant le fait que le bruit ne dégradait quasiment pas les résultats.

En somme, nous proposons dans cette thèse une méthode novatrice qui enrichit l'information de classe en proposant un découpage en sous-classes. Que ce soit dans une but de caractérisation ou pour prétraiter des données, notre algorithme permet d'utiliser pleinement l'information de classe conjointement à l'information contenue dans les descripteurs.

Dans la partie suivante, nous illustrons la richesse de notre approche en l'utilisant dans de nombreuses applications allant de la construction de prototypes à l'amélioration d'algorithmes d'apprentissage existants.

Troisième partie

Expérimentations

Chapitre 1

Données artificielles

Dans ce chapitre, nous présentons les expérimentations réalisées sur des jeux de données artificielles en deux dimensions. Ces données ont l'avantage de permettre une visualisation simple des résultats obtenus. Le caractère artificiel des jeux de données nous permet de mettre en évidence certains aspects de la méthode.

Ce chapitre est découpé en cinq parties. Dans un premier temps, nous illustrons la découverte automatique de sous-classes pertinentes par notre approche ; et cela même dans des situations relativement complexes. Par la suite, nous nous intéressons au sens que revêt le calcul de la typicalité dans les cas où un découpage en sous-groupes est disponible. Puis, dans la section suivante, nous montrons l'avantage certain que les sous-classes offrent pour le calcul de prototypes. Ensuite, nous proposons et testons un algorithme simple de classification qui se base sur les prototypes des classes pour étiqueter les nouveaux exemples. Enfin, avant de conclure le chapitre, nous mettons plus particulièrement en avant un des aspects de notre approche qui est la détection d'exemples exceptionnels.

1.1 Détection efficace de sous-classes

Cette expérimentation a été menée dans le cadre d'un article soumis à la conférence *RJCIA2009* (Forest & Rifqi, 2009). L'objectif est de valider de façon empirique la détection correcte de comportements par notre méthode de calcul de sous-classes.

1.1.1 Génération des données

Ainsi, nous avons construit un jeu de données où une classe est homogène et deux classes ne le sont pas ; c'est-à-dire constituées de sous-groupes. Pour créer ce jeu de données artificielles, nous avons utilisé le générateur aléatoire du logiciel Matlab et constitué 9 groupes de points répartis au hasard sur deux dimensions selon des distributions

gaussiennes autour de 9 centres. Les points ont été rangés dans trois catégories (voir la figure 1.1) : aux quatre coins se trouvent les exemples de la classe 1, le groupe central pour la classe 2 et les quatre groupes se trouvant sur les cotés pour la classe 3. Chacun des groupes est constitué de 50 individus à l'exception du groupe central qui en compte 100. Par conséquent les classes 1 et 3 ont une cardinalité de 200 exemples et la classe 2 a une cardinalité de 100 exemples.

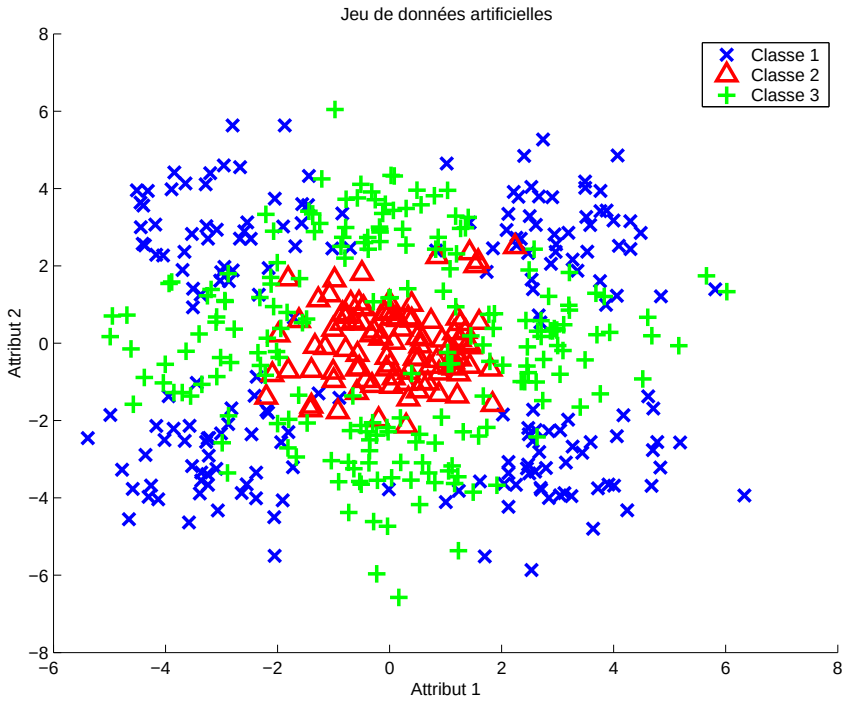

FIGURE 1.1 – Base de données artificielles partitionnée en 3 classes dont deux sont hétérogènes.

1.1.2 Tolérance fixe

En supposant le paramètre de tolérance fixé à n, la première étape de notre approche est le calcul de la matrice de distances pour tous les individus. Ensuite, à chaque exemple est associée une liste des distances qui le séparent des autres. En parcourant cette liste et en s'arrêtant au $n^{ième}$ exemple qui n'est pas de sa classe on obtient le rayon de son *cercle d'influence*, c'est-à-dire le cercle qui contient les exemples auxquels il se relie.

Sur la figure 1.2, on peut voir l'ensemble des liens construits sur le jeu de données artificielles pour une tolérance nulle, c'est-à-dire $n = 0$. Pour ne pas encombrer la figure, tous les exemples sont représentés par des cercles noirs. Les liens sont représentés sous la

forme de lignes dont la couleur correspond à la classe des exemples reliés. A ce stade de l'algorithme, on dispose donc d'une matrice d'adjacence.

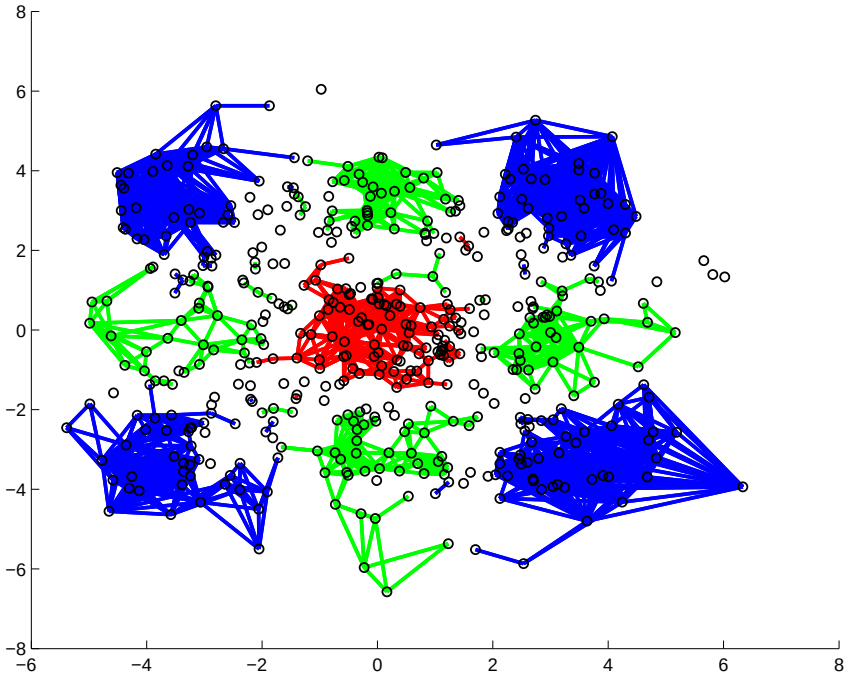

FIGURE 1.2 – Liens entre individus pour un paramètre de tolérance égal à 0.

L'étape suivante est l'affectation des sous-classes qui est réalisée en parcourant les différents graphes et chaque graphe constituant une sous-classe. S'il existe un chemin qui relie deux exemples, alors ils sont de la même sous-classe. La figure 1.3 illustre les sous-classes découvertes.

Notre approche a mis en évidence sur ce jeu de données 98 sous-classes distinctes. Ce nombre peut paraître élevé. Il faut cependant bien noter que le paramètre de tolérance est ici égal à 0, ce qui maximise la segmentation en stoppant la progression des cercles d'influence des exemples. De plus, ce jeu de données contient beaucoup de zones de chevauchement entre classes qui favorisent l'émergence des sous-classes de très petite taille.

Nous avons tracé l'histogramme des cardinalités des sous-classes sur la figure 1.4. On remarque que sur les 98 sous-classes découvertes, seulement 10 d'entre elles dépassent la taille de 5 exemples (soit 1% du nombre total d'exemples de la base). De plus, 9 sous-classes sortent radicalement du lot avec une cardinalité minimum d'une trentaine d'exemples. Ces neufs sous-groupes sont visibles sur les figures 1.2 et 1.3. Il s'agit du cœur des 9 groupes générés initialement.

Ainsi, notre objectif est atteint : notre méthode a détecté avec succès les différents

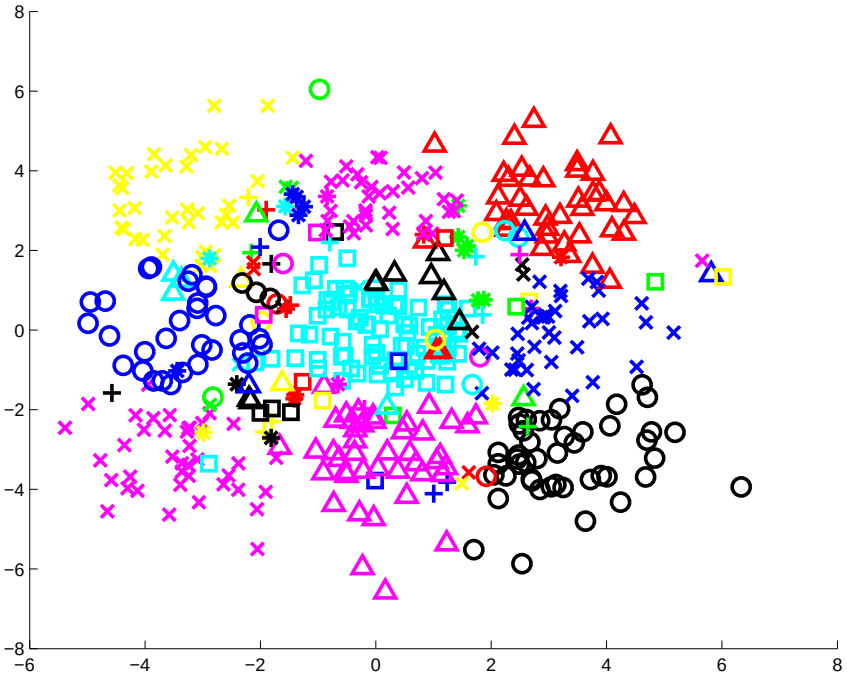

FIGURE 1.3 – Découpage du jeu de données en 98 sous-classes.

comportements qui étaient présents dans les trois classes du jeu de données.

1.1.3 Etude de la tolérance

Dans l'expérience précédente nous avons délibérément fixé le paramètre de tolérance à zéro pour illustrer un cas extrême. Pour comprendre l'influence de ce paramètre nous avons étudié l'impact des variations de tolérance sur le nombre de sous-classe découvertes et sur leur taille. Pour cela nous avons construit les sous-classes sur le jeu de données des 9 groupes, en faisant varier progressivement la tolérance de 0 à 5. Le tableau 1.1 (page 74) regroupe les résultats que nous avons obtenus. Pour chaque valeur de tolérance (première colonne) et pour chaque classe (deuxième colonne) le nombre de sous-classes est donné, suivi de la cardinalité des 5 sous-groupes les plus volumineux.

La première constatation est que, comme on pouvait s'y attendre, plus la tolérance est élevée et plus le nombre de sous-classes découvertes est faible : les exemples se relient plus facilement les uns aux autres et les réseaux sont par conséquent plus étendus. Avec une tolérance à 0 on a 98 sous-classes alors qu'en la réglant la tolérance à 21 on ne segmente pas.

Le deuxième fait qui mérite d'être relevé est que le réglage optimal de la tolérance semble être la valeur 2. En effet, c'est là qu'on obtient une classe 2 peu segmentée et des

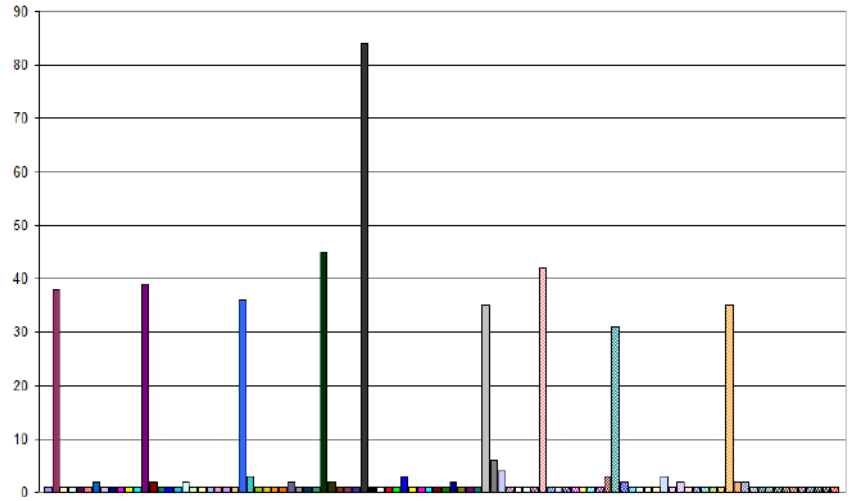

FIGURE 1.4 – Découpage du jeu de données en 98 sous-classes. 9 se distinguent par une cardinalité élevée.

classes 1 et 3 correctement segmentées en 4 sous-groupes de tailles importantes. Pour une tolérance égale à 3 on voit que la classe 3 est segmentée en trois sous-groupes majeurs au lieu de 4 attendus : les réseaux se sont suffisamment étendus pour traverser les zones de chevauchement et relier des groupes distants.

En poussant plus loin l'analyse, on peut observer qu'il semble y avoir non pas un réglage global de la tolérance mais plutôt une valeur optimale *pour chaque classe* du jeu de données. En effet, en passant d'une valeur de tolérance de 2 à 3, deux importants sous-groupes de la classe 3 se relient alors que ce phénomène ne s'observe pas pour la classe 1, même pour une tolérance de 10. Ces différences de réactions au paramètre de tolérance entre la classe 1 et la classe 2 s'expliquent par le fait que les différents groupes qui les composent ne sont pas séparés de la même façon et cela pour deux raisons. Si on englobait les données dans un carré, les groupes qui constituent la classe 3 seraient situés sur les côtés de celui-ci alors que les groupes de la classe 1 seraient situés aux quatre coins. Les distances inter-groupes sont donc plus grandes pour la première classe. Mais l'élément vraiment déterminant est que les groupes de la classe 1 sont plus *séparés* que ceux de la classe 3 ; c'est-à-dire qu'entre deux groupes il y a plus d'individus d'une autre classe : la tolérance n'est pas assez forte pour permettre aux individus de passer cette barrière et de se lier.

Ainsi, autant sur des données artificielles en deux dimensions il est aisé d'apprécier l'impact du paramètre de tolérance, autant sur des données plus complexes ce réglage nécessitera d'être prudent.

Valeur de tolérance	Classes	Nombre de sous-classes	Cardinalité des cinq principales				
0	1	39	45	39	38	36	3
	2	14	84	3	2	1	1
	3	45	42	35	35	31	6
1	1	21	50	44	43	40	3
	2	13	85	3	2	1	1
	3	25	48	45	41	40	3
2	1	13	50	48	46	45	2
	2	7	92	3	1	1	1
	3	17	50	49	44	42	2
3	1	10	50	49	48	46	2
	2	5	94	3	1	1	1
	3	15	94	49	43	2	2
4	1	8	50	49	49	47	2
	2	4	95	3	1	1	
	3	10	146	45	2	1	1
5	1	8	50	49	49	47	2
	2	2	95	5			
	3	7	193	2	1	1	1
10	1	5	51	50	49	49	1
	2	1	100				
	3	2	199	1			

TABLE 1.1 – Tableau du nombre de sous-classes et de leur cardinalité obtenus sur le jeu de données des 9 groupes en faisant varier le paramètre de tolérance.

1.2 Typicalités et sous-classes

Comme nous l'avons détaillé dans la partie 1.3, la typicalité est un indice qui permet de savoir si un exemple donné est caractéristique de sa classe - c'est-à-dire s'il ressemble à ceux de sa catégorie tout en étant différent des exemples des autres catégories. Ainsi, la typicalité d'un exemple dépend intrinsèquement de sa place dans l'espace des données relativement à la place des autres. Par conséquent, si l'affectation des classes change, la typicalité change.

Un autre point important, comme nous l'avons souligné dans la partie 2.2, est que la typicalité est élevée pour les exemples appartenant à des classes compactes, et basse sur ceux de classes éclatées en plusieurs groupes. En effet, il est normal de ne pas trouver *un* comportement typique si *plusieurs* comportements différents cohabitent. C'est pourquoi nous proposons un enrichissement du calcul des valeurs de typicalité en se basant non plus sur les informations de classes mais plutôt sur un découpage en sous-classes.

Dans la suite de cette partie nous commencerons par illustrer le fait que la typicalité sur des classes éclatées donne de mauvais résultats. Puis, dans un second temps, nous verrons plusieurs façons d'exploiter un découpage des classes pour calculer les typicalités différemment. Pour chacune de ces possibilités nous mènerons une série de tests pour tenter de les caractériser.

Nous allons illustrer nos propos sur un jeu de données artificielles. Celui-ci se compose de deux classes de 60 exemples chacun décrits par deux attributs. Comme on le voit sur la figure 1.5 qui représente les données, la première classe est hétérogène alors que la seconde est compacte. On suppose également qu'en plus de l'information d'affectation en classes, on dispose d'un découpage parfait en sous-classes. 3 sous-classes sont donc identifiées : la classe 2 n'est pas divisée et la classe 1 est segmentée en deux parties, représentées par des carrés et des cercles sur la figure.

Pour les calculs de typicalité nous avons utilisé des mesures de ressemblance interne et de dissimilarité externe basées sur des distances. Ainsi étant donné un univers partitionné en m classes : $X = \{C_1, \ldots, C_m\}$. Les formules de calculs de la ressemblance interne, de la dissimilarité externe et de la typicalité d'un exemple x sont :

$$\begin{aligned} R(x) &= Moy_{\substack{y \in C_k \\ x \neq x}}(1 - d(x,y)) \\ D(x) &= Moy_{y \notin C_k} d(x,y) \\ T(x) &= \frac{R(x).D(x)}{R(x).D(x) + (1 - R(x))(1 - D(x))} \end{aligned}$$

où Moy désigne la moyenne arithmétique usuelle, d est la distance euclidienne normalisée, et l'opérateur d'agrégation utilisé pour le calcul de la typicalité est la somme

symétrique (Silvert, 1979), opérateur dit « à attitude variable » qui a l'avantage d'être de type conjonctif pour des basses valeurs (inférieures à 0.5) : la valeur de l'agrégation est proche du minimum de la ressemblance et de la dissimilarité et de type disjonctif dès que l'une des valeurs est élevée (supérieure à 0.5) : la valeur de l'agrégation est proche du maximum de la ressemblance et de la dissimilarité.

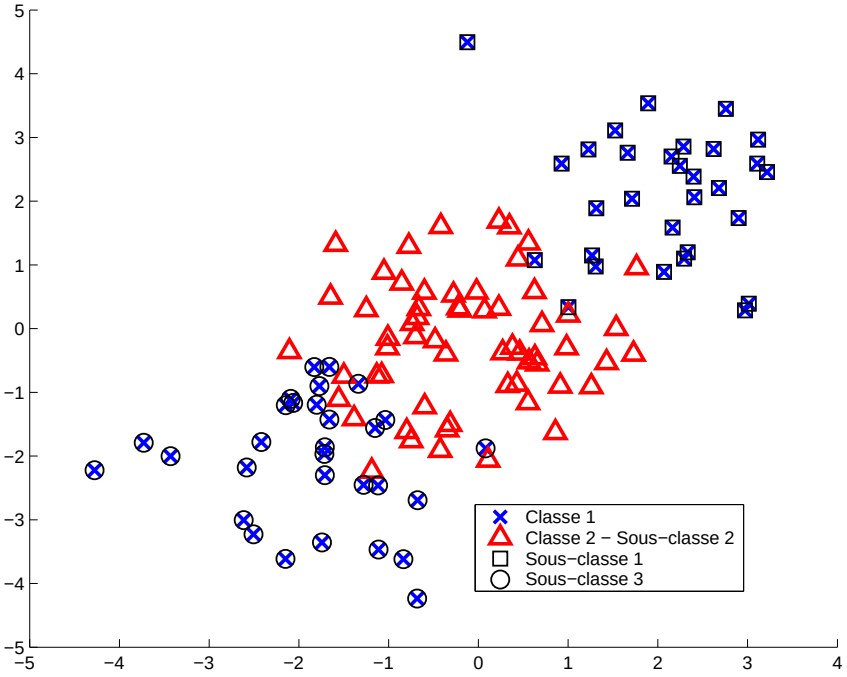

FIGURE 1.5 – Jeux de données artificielles. Deux classes et trois sous-classes sont identifiées.

Les scores de typicalité des exemples de ce jeu de données sont calculés en nous appuyant sur l'information d'appartenance aux classes - c'est-à-dire le calcul traditionnel de la typicalité. Il est possible de représenter graphiquement les scores obtenus en dessinant les lignes de niveaux de typicalité. La figure 1.6 montre ces résultats : les zones de l'espace des données sont représentées par une couleur qui est froide (bleue) si la typicalité est très basse (proche de 0) et de plus en plus chaude (rouge) selon que la typicalité augmente. On peut donc très rapidement voir les zones typiques et les zones non typiques.

En analysant les résultats, on constate que la classe 2 (au centre) admet des scores de typicalité élevés compris entre 0.68 et 0.86 ; les valeurs les plus fortes étant atteintes par les exemples les plus au centre de la classe. Ces scores permettent de conclure que *ces exemples centraux sont typiques de leur classe*. Pour ce qui est de la classe 1, les scores sont compris entre 0.2 et 0.42 : on ne trouve pas d'exemple typique pour cette classe - ce qui est troublant étant donné la présence des deux groupes clairement identifiés. *Ainsi, les*

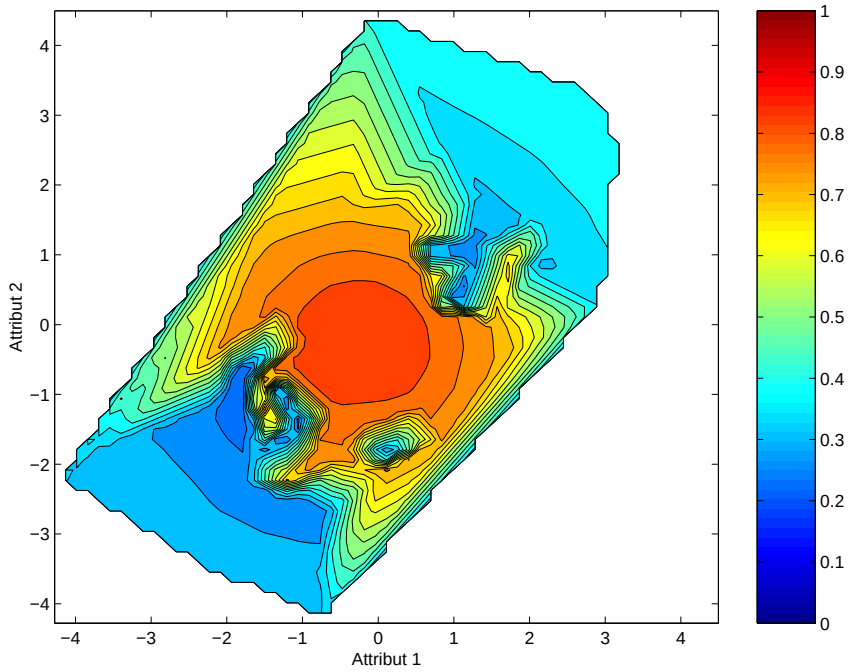

FIGURE 1.6 – Gradient des scores de typicalité obtenus par le calcul traditionnel.

typicalités calculées de façon traditionnelle sur des classes non homogènes ne permettent pas de mettre en évidence les différents comportements.

Pour prendre en compte les classes non homogènes, il faut se servir du découpage en sous-classes. La typicalité traditionnelle calcule, pour un exemple donné, sa ressemblance interne sur les exemples de sa classe (*les amis*) et sa dissimilarité externe sur les exemples des autres classes (*les ennemis*). Dans le cas où une information de sous-classes vient s'ajouter, nous avons proposé de redéfinir ces deux notions en proposant trois approches différentes. Ainsi, un exemple aura comme vision :

1. Approche hostile : mes amis sont ceux de ma sous-classe. Mes ennemis sont ceux des autres sous-classes.

2. Approche neutre : mes amis sont ceux de ma sous-classe. Mes ennemis sont ceux des autres classes.

3. Approche étrange : mes amis sont ceux de ma classe. Mes ennemis sont ceux des autres sous-classes.

Pour chacune de ces trois approches nous avons calculé les scores de typicalité et dessiné les résultats sous forme de ligne de niveaux.

1.2.1 Approche hostile

Nous avons qualifié cette façon de se servir des sous-classes d'hostile car des exemples avec lequels on cherche des ressemblances dans le calcul de la typicalité traditionnel (des amis) sont à présent considérés comme ennemis. Dans cette approche, on descend au niveau des sous-classes et on « oublie » l'information de classe.

Pour notre jeu de données, les scores que nous avons obtenus sont sur la figure 1.7. La constatation la plus importante est qu'avec cette approche les trois principaux comportements du jeu de données sont bien restitués : on observe en effet trois pics aux alentours de 0.8 centrés sur les trois groupes. De plus, les zones de chevauchement des classes ont des scores plus bas ce qui est satisfaisant. Cette approche donne donc de bons résultats. Seul désavantage, du fait d'être entourée de part et d'autre, la classe centrale admet des scores de typicalité un peu moins forts que les deux groupes de la classe 1.

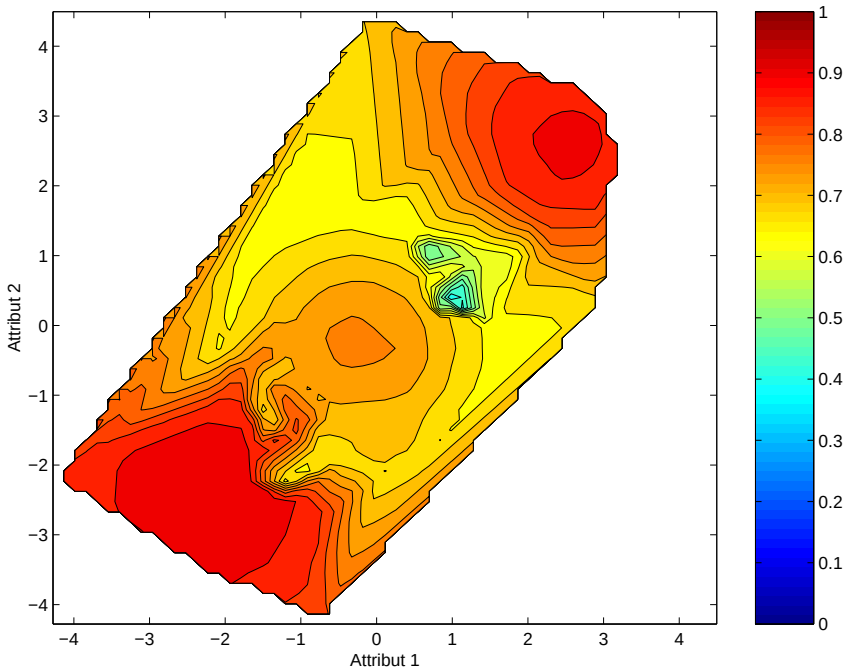

FIGURE 1.7 – Lignes de niveaux des scores de typicalité pour l'approche hostile.

1.2.2 Approche neutre

L'approche neutre utilise l'information de sous-classes pour restreindre ses amis - les ennemis restant les même. On a donc tout un ensemble d'exemples qui ne sont plus pris en compte dans le calcul de la typicalité. Les résultats obtenus sur notre jeu de données se trouvent sur la figure 1.8. Une fois encore les trois différents comportements

sont correctement rendus. Cependant, le désavantage de cette approche est que les scores de typicalité sont relativement hauts (> 0.75) sur presque tout l'espace. On a donc plus de mal à faire la différence entre les exemples typiques et ceux qui ne le sont pas beaucoup.

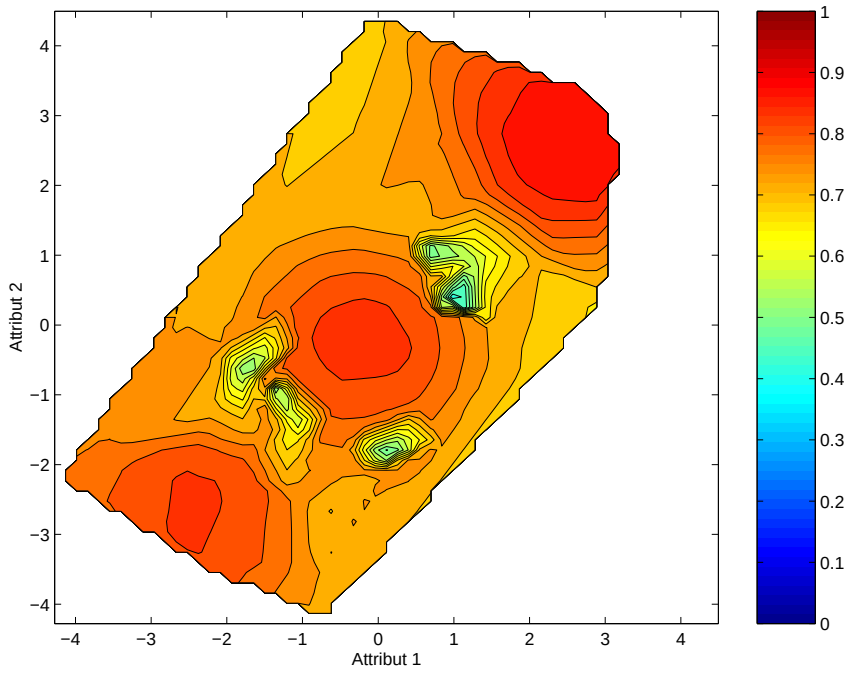

FIGURE 1.8 – Lignes de niveaux des scores de typicalité pour l'approche neutre.

1.2.3 Approche étrange

Contrairement à la méthode neutre, l'approche étrange utilise certains exemples à la fois comme amis et comme ennemis. Les résultats sont représentés sur la figure 1.9. Bien qu'étant légèrement meilleurs qu'un calcul de typicalité traditionnel, les résultats sont mauvais et on ne retrouve pas les deux comportements de la classe 1.

1.2.4 Conclusion

Dans cette partie nous avons montré que le calcul de la typicalité tel qu'il est traditionnellement effectué ne donne pas de résultat satisfaisant lorsqu'il s'agit de caractériser des classes non homogènes. Nous avons alors proposé de redéfinir ce calcul en utilisant un découpage en sous-classes. Pour cela nous avons imaginé trois façons d'exploiter l'information de sous-classes et deux d'entre elles ont donné de bons résultats. Les méthodes à base de typicalité peuvent donc être améliorées si un découpage en sous-classe est disponible.

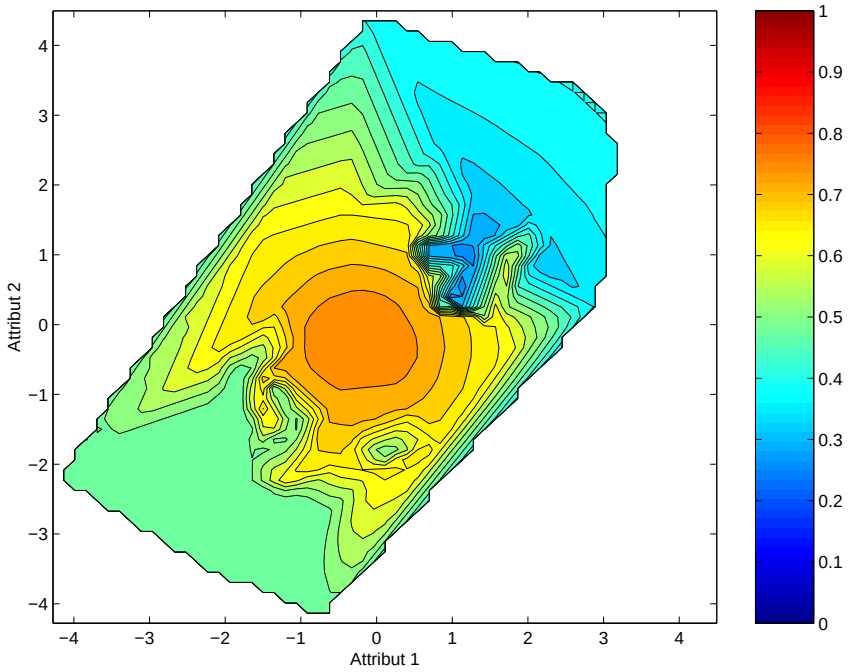

FIGURE 1.9 – Lignes de niveaux des scores de typicalité pour l'approche étrange.

1.3 Amélioration de la construction de prototypes flous

Nous avons montré, au début de ce chapitre, que la méthode de découverte automatique de sous-classes que nous proposons est efficace. Puis, nous avons montré que les méthodes à base de typicalité peuvent tirer un avantage certain des découpages en sous-classes et nous avons validé deux propositions de calcul de la typicalité en présence de sous-classes. Dans cette partie nous allons relier ces deux idées et montrer que notre méthode améliore de façon significative la construction de prototype de classe - méthode qui se base sur la typicalité.

Pour cette expérience nous utilisons le jeu de données présenté dans la partie 1.1 et illustré sur la figure 1.1, page 70. Celui-ci se compose de trois classes, une centrale et deux autres éclatées en quatre groupes chacune.

Sur ce jeu de données nous avons, dans un premier temps, extrait les prototypes en utilisant un algorithme à base de typicalité sur les classes. Puis, dans un second temps, nous avons proposé une nouvelle méthode d'extraction de prototypes qui s'appuie sur la découverte automatique de sous-classes de notre méthode. Le but de ces deux expériences est de comparer les approches et de montrer que la nôtre permet de trouver des prototypes doté d'un plus grand pouvoir de représentativité et donc de meilleure qualité.

1.3.1 Prototypes par la typicalité usuelle

Dans la première partie de notre expérimentation nous avons calculé les scores de typicalité de manière traditionnelle pour l'ensemble des exemples de la base ; c'est-à-dire que la ressemblance interne d'un exemple donné est calculée en prenant en compte les exemples de sa classe et sa dissimilarité externe les exemples des autres classes. Puis nous avons extrait les prototypes en choisissant, pour chaque classe, l'exemple qui maximisait le score de typicalité. Nous avons donc obtenu trois prototypes différents. La figure 1.10 montre les résultats obtenus : l'image en haut représente les lignes de niveaux de typicalité sur l'ensemble des données. L'image du bas met en évidence les trois prototypes en encadrant les exemples qui maximisent les scores de typicalité pour chaque classe.

On constate, en regardant la figure de gauche, que les scores de typicalité sont satisfaisants uniquement sur la classe centrale ; les deux autres classes n'ont aucun exemple typique ce qui est problématique pour la caractérisation. Une fois encore, nous observons que les classes non homogènes ne sont pas bien prises en compte par le calcul usuel de typicalité. De plus, pour ce qui est des prototypes extraits, même si la classe centrale a un bon représentant, les classes 1 et 3 sont quant à elles très mal représentées. En effet, dans le cas de la classe 1, le prototype est un exemple très excentré et qui pourrait quasiment être considéré comme une exception. Ce résultat s'explique par le fait que cet exemple a une forte dissimilarité. A l'inverse, le prototype de la classe 3 est un exemple qui est central pour sa classe : il a une forte ressemblance interne.

On peut donc conclure que les prototypes mis en avant par la typicalité usuelle ne sont pas adaptés.

1.3.2 Prototypes basés sur le découpage des classes

Pour mettre en avant des prototypes qui correspondent à la réalité des données nous avons proposé une méthode en plusieurs étapes - la méthode des prototypes multiples :

1. On lance une découverte des sous-classes par notre méthode de réseaux d'amis.
2. On identifie les sous-classes représentatives. Ce sont les sous-classes dont le nombre d'exemples dépasse un certain seuil passé en paramètre.
3. On calcule les scores de typicalité en utilisant l'approche *hostile* (définie dans la partie 1.2.1, page 78) en ne prenant en compte que les sous-classes significatives.
4. Soit une classe C se décomposant en sous-classes $C = sc_1 \cup \ldots \cup sc_n$ on définit le prototype P de la classe C comme $P = \{p_1, \ldots, p_n\}$ où p_i est l'exemple de la sous-classe sc_i qui maximise la typicalité pour sa sous-classe.

Le choix de mettre de côté certains exemples de la base de données et de ne pas s'en servir pour le calcul des typicalités peut être discuté. Nous pensons pourtant que lors de la construction de prototypes il est important de se concentrer exclusivement sur les grandes tendances des données en ne tenant pas compte des tendances minoritaires.

En pratique, pour le jeu de données dont il est question ici, nous avons fait un découpage en sous-classes en réglant le paramètre de tolérance à 0. Ainsi, comme dans la partie 1.1, nous avons découvert 98 sous-classes (dont 9 majeures). Pour ce qui est du filtrage des sous-classes nous avons fixé une taille minimale de 10 exemples. Notre jeu de données filtrées compte donc 385 exemples sur les 500 de départ. Nous avons calculé les scores de typicalité en utilisant l'approche *hostile* sur le jeu de données filtrées. Les gradients de typicalité obtenus sont visibles sur la figure 1.11 en haut. Enfin, nous avons extrait les prototypes du jeu de données. Ils sont mis en relief sur la figure 1.11 en bas.

L'analyse des résultats montre que notre méthode fournit des prototypes qui rendent correctement compte des tendances du jeu de données. La classe 2, qui est centrale, a pour prototype un unique exemple situé au sein de la classe : il est très ressemblant aux exemples de sa classe et très dissimilaire de ceux des autres classes. Les classes 1 et 3 ont pour prototype un ensemble de quatre exemples chacune. Ces quatre exemples se situent tous au centre d'un des groupes de la classe, ce qui reflète parfaitement les tendances de la classe.

Le « pouvoir de description » d'un prototype est donnée par sa typicalité. Le tableau 1.2 dresse un comparatif des typicalités. La seconde colonne liste les scores de typicalité des prototypes qui n'utilisent pas les sous-classes alors que la troisième colonne donne la moyenne des typicalités obtenues pour les prototypes multiples. Nous avons ajouté entre parenthèses les quatre scores des exemples qui forment les prototypes des classes 1 et 3.

Classe	Approche traditionnelle	Notre approche
1	0.43	0.96 (0.97, 0.96, 0.96, 0.95)
2	0.89	0.92
3	0.56	0.95 (0.97, 0.94, 0.95, 0.95)

TABLE 1.2 – Comparaison des scores de typicalité entre l'approche traditionnelle et notre approche.

On constate à la vue de ces résultats que, dans tous les cas, nos prototypes surclassent ceux obtenus par l'approche traditionnelle. Nous pouvons donc conclure que, sur ce jeu de données, contrairement à la méthode usuelle, notre approche des prototypes multiples est meilleure.

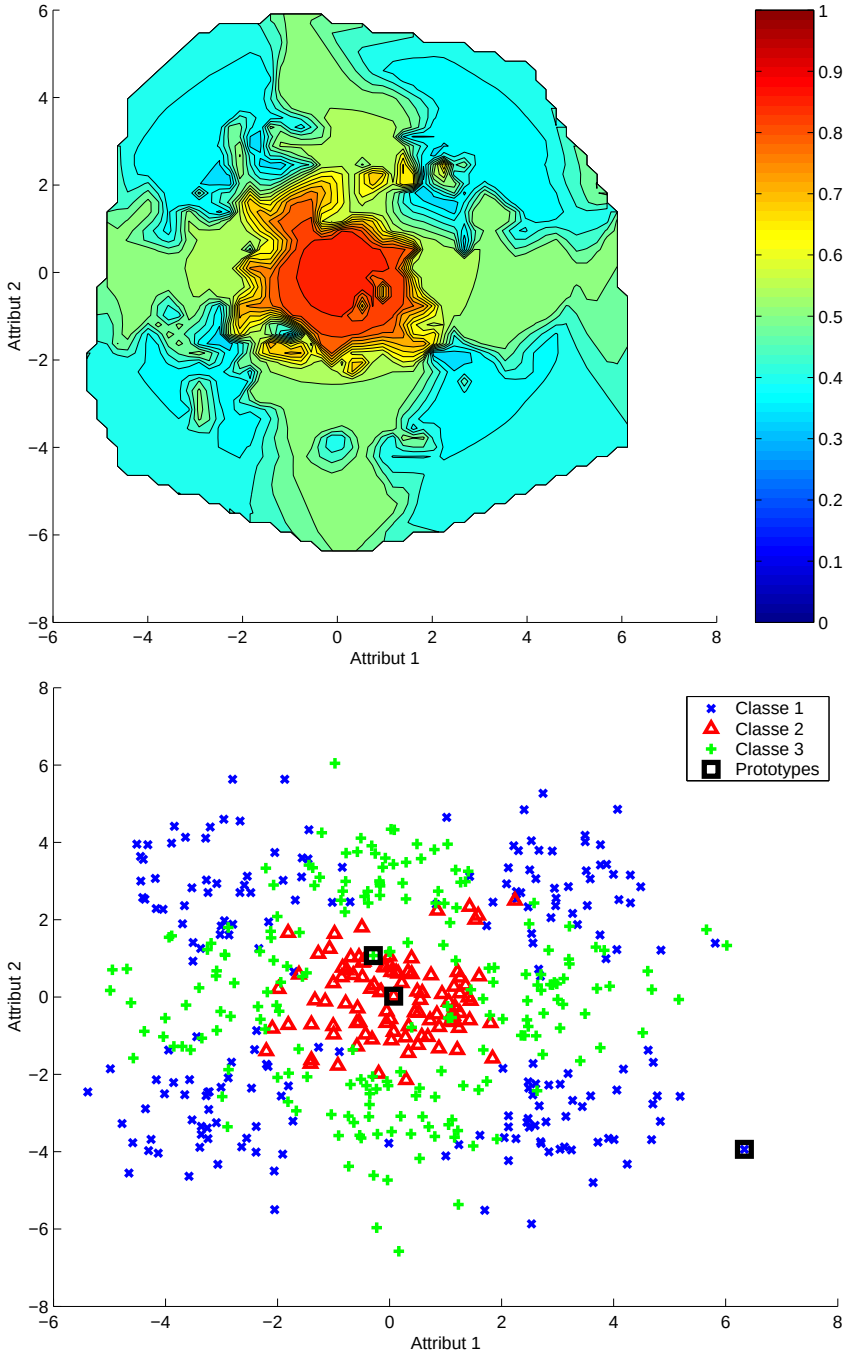

FIGURE 1.10 – En haut : lignes de niveaux des scores de typicalité par l'approche traditionnelle. En bas : prototypes des trois classes.

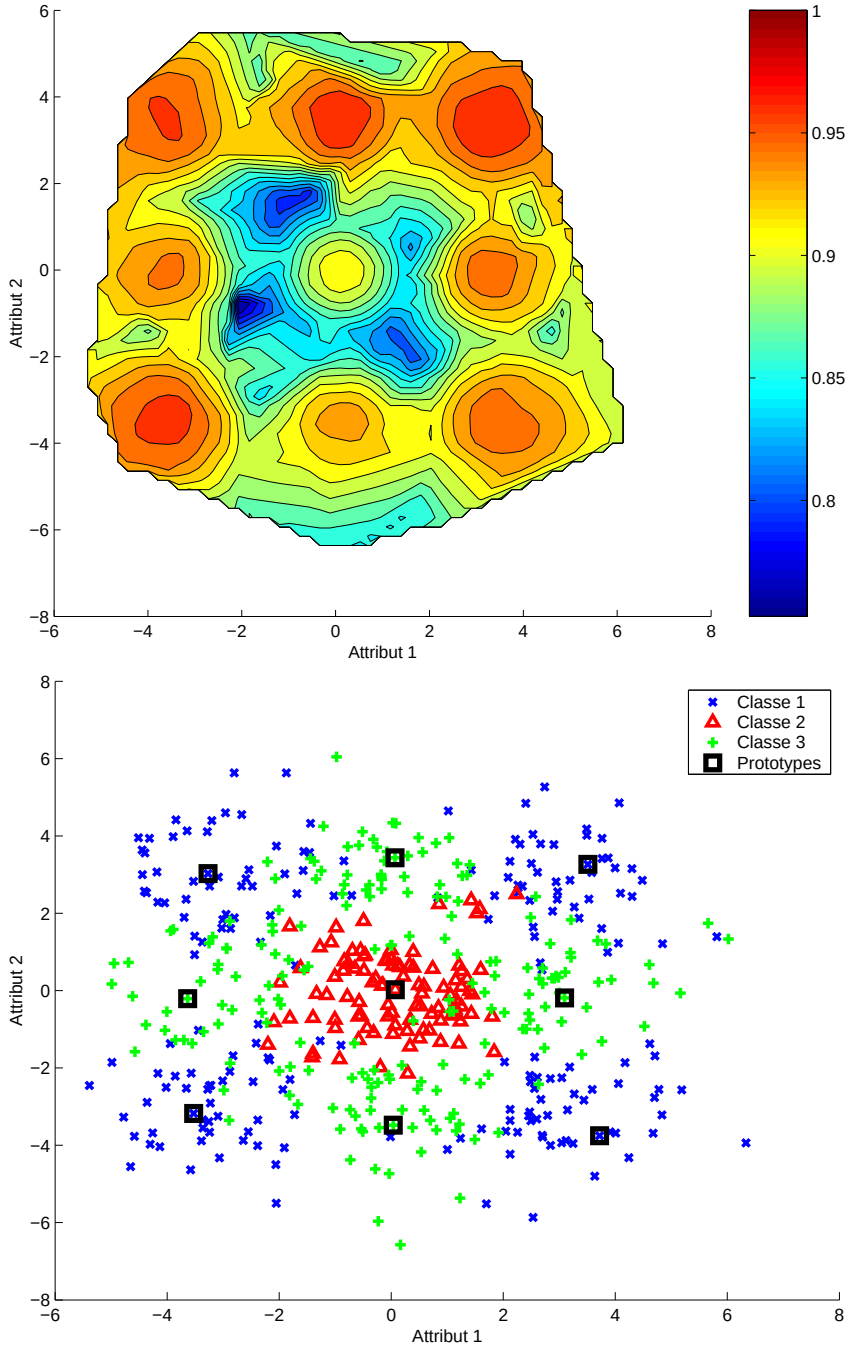

FIGURE 1.11 – En haut : lignes de niveaux des scores de typicalité par l'approche hostile sur les données filtrées. En bas : les prototypes multiples des trois classes.

1.4 Synthèse

Dans ce chapitre nous avons validé sur un jeu de données artificielles l'approche de segmentation de classes qui est le cœur de notre travail de thèse. Puis, nous avons montré que, par le biais des découpages en sous-classes, il était possible d'enrichir les méthodes à base de typicalité en utilisant conjointement l'appartenance d'un exemple à sa classe et à sa sous-classe. Nous avons proposé deux méthodes innovantes pour le calcul de typicalité sur classes non homogènes et les gains qu'elles apportent ont été mis en avant. Enfin, dans la dernière partie nous avons combiné la découverte de sous-classes et le calcul de typicalité pour proposer un moyen de construire des prototypes capables de restituer correctement les différents comportements d'une classe. Cette approche a été comparée à une approche plus traditionnelle sur un jeu de données artificielles. Les avantages qu'elle apporte sont indéniables.

Chapitre 2

Données réelles

Ce chapitre regroupe les expérimentations que nous avons effectuées sur des jeux de données étiquetées réelles. Notre thématique de recherche est la caractérisation de classes et notamment celle qui traite le cas des classes non homogènes. Or, les données issues de vrais problèmes ont beaucoup de chances de contenir des classes qui englobent des comportements différents. Nous pensons donc que nos travaux peuvent être utiles.

Contrairement aux données artificielles présentées au chapitre précédent, les données que nous traitons dans ce chapitre sont toutes décrites par un grand nombre de dimensions. Il n'y a donc pas la possibilité de visualiser les résultats sous forme de graphiques comme nous le faisions. La prise en compte de données réelles a aussi été le moyen de tester nos méthodes sur des grands jeux de données.

Dans ce chapitre, nous allons détailler quatre expériences différentes. Nous commençons par un travail d'aide à la caractérisation de prescription médicale que nous avons effectué en association avec une entreprise de l'industrie pharmaceutique. Puis, nous détaillons les tests de construction de prototypes multiples menée sur les données de l'UCI Machine Learning Data Repository (UCI, 2009). Dans une troisième partie nous nous intéressons à un résumé d'une base d'images de paysages. Enfin nous terminons par une étude sur la caractérisation d'écriture manuscrite.

2.1 Caractérisation de la prescription de médicaments

2.1.1 Contexte de l'étude et présentation des données

Cette expérience a pour but l'analyse et la caractérisation du comportement de prescription d'un échantillon de médecins vis-à-vis d'un certain nombre de médicaments. Nous avons réalisé ce travail en collaboration avec l'entreprise ARVEM France qui a financé

notre thèse.

L'entreprise ARVEM est une PME qui compte un peu plus de 400 personnes. Son activité principale est la « promotion médicale ». Elle consiste à démarcher les médecins pour le compte d'un laboratoire pharmaceutique et à leur présenter certains médicaments. On appelle *délégué à l'information médicale* ou encore *visiteur médical*, les commerciaux qui vont voir les praticiens sur leur lieu de travail. Chaque visiteur a un secteur géographique bien déterminé qui regroupe en moyenne plusieurs milliers de médecins. Il a en sa possession la liste de tous les médecins qui travaillent sur son secteur. L'efficacité du visiteur médical se mesure grâce à l'évolution des chiffres de ventes des médicaments qu'il a présentés.

Dans le but d'aider les laboratoires à mieux cibler les médecins susceptibles d'être réceptifs à leur argumentaire de vente, la société Cegedim, acteur très présent du marketing pharmaceutique, vend de nombreuses bases de données sur les médecins et sur leur activité. Parmi celles-ci on trouve notamment l'indice *Icomed*. Cet indicateur est une auto-évaluation du médecin sur sa prescription : un panel de médecins volontaires reçoit un questionnaire dans lequel on leur demande de noter sur une échelle de 0 à 3 leur prescription pour environ 500 médicaments avec la possibilité de ne pas se prononcer. Le tableau ci-dessous synthétise les choix possibles :

0	Je **ne prescris pas**
+	Je prescris **occasionnellement** ou pour une indication particulière
++	Je prescris **régulièrement**
+++	Je prescris **beaucoup**

Pour effectuer notre travail nous avons eu accès à une base de données comportant l'auto-évaluation d'un peu plus de 2300 médecins sur 7 médicaments différents.

Il existe une autre source de données dont nous nous sommes servis : le logiciel TEAMS. Ce logiciel permet aux laboratoires pharmaceutiques d'accéder à des informations sur les médecins. Il y a à la fois des informations « administratives » (âge, sexe, adresse) et des informations « qualitatives ». Ces dernières ne sont généralement pas commentées, et, malgré tous nos efforts, nous n'avons jamais obtenu de précision quant aux sens ou aux calculs de ces indices. On trouve notamment : une cotation du lieu d'exercice du médecin, un indice de la sollicitation de la part des visiteurs médicaux (la pression de visite à l'adresse), deux coordonnées X et Y et un « potentiel » qui semble quantifier à quel point le médecin est « intéressant ». Bien qu'il y ait une centaine de descripteurs, une très grande majorité n'a aucun intérêt pour notre analyse (comme par exemple le numéro d'inscription à l'ordre des médecins, le prénom ou le code de la zone géographique) et on trouve une grande redondance dans les données (exemple : les jours de réception sur

rendez-vous décrits de trois façons différentes).

Nous avons ainsi combiné ces deux sources et effectué un long travail de sélection et de nettoyage des données pour produire une base de 2326 exemples (médecins) décrits par 9 attributs numériques ou binaires. Ces descripteurs sont :

1. le sexe du médecin,
2. son âge,
3. la limite de visite : une note entre 0 et 9. Plus elle est haute et moins le médecin reçoit de visiteurs médicaux,
4. la pression de visites à l'adresse : le nombre moyen de visites de délégués médicaux reçus à l'adresse en un mois,
5. le pourcentage de réception sur rendez-vous,
6. la cotation à l'adresse : une note entre 0 et 9 qui renseigne sur la « qualité » du lieu d'exercice du médecin.
7. le potentiel : un indice compris entre 0 et 1000 qui qualifie la prescription en général du médecin,
8. les coordonnées X et Y : aucune information supplémentaire n'est disponible sur ces descripteurs.

Pour ce qui est des classes, nous disposons de 7 affectations différentes que nous avons nommées Produit1 à Produit7. Le tableau ci-dessous donne les répartitions des 2326 exemples où chaque ligne est un produit différent et les colonnes renseignent sur le nombre de médecins qui appartiennent à une certaine classe.

Produit	0	+	++	+++
1	0	619	740	967
2	79	526	1096	625
3	181	676	999	470
4	179	732	1019	396
5	0	404	895	1027
6	746	761	678	141
7	84	267	1609	376

On constate que les classes sont très différentes les unes des autres : par exemple les produits 1 et 5 sont très prescrits, alors que le produit 6 ne l'est pas beaucoup. De plus, nous sommes bien conscients que cet indice Icomed, qui nous sert de classe, est très subjectif car issu d'une auto-évaluation. Les médecins sont bien conscients que, suivant la catégorie dans laquelle ils vont se ranger, les démarcheurs viendront plus ou moins les solliciter.

2.1.2 Etudes préliminaires : Apprentissage de l'indice ICOMED

Sur ces données nous avons directement appliqué des algorithmes traditionnels d'apprentissage. Nous allons, dans cette partie, décrire notre tentative d'apprendre un modèle en utilisant les arbres de décision. Pour cela nous avons utilisé le logiciel Weka de l'université de Waikato (Université de Waikato, 2009) qui propose une implémentation très complète de l'algorithme C4.5 de Quinlan (1993). Notre protocole a été le suivant : pour les 7 classes que nous avons à notre disposition nous construisons un arbre sur 2/3 des données et nous testons cet arbre sur le tiers restant. Nous faisons cela 30 fois en notant à chaque fois le taux de bonnes classifications et le nombre de feuilles de l'arbre obtenues. Au final, nous faisons la moyenne des 30 exécutions. Les paramètres de construction de l'arbre sont : un indice de confiance de 0.25 pour l'élagage et un nombre minimum de 2 exemples par feuille. Les résultats que nous avons obtenus se trouvent dans le tableau 2.1. Il y a un produit par ligne. La première colonne donne le taux moyen de bonnes classifications alors que la seconde colonne renseigne sur le nombre moyen de feuilles qu'a l'arbre. Enfin la dernière colonne donne le taux de reconnaissance qu'on peut obtenir en classant tous les exemples dans la classe majoritaire.

Produit	Taux de bonne classification moyen	Nombre moyen de feuilles	Classification majoritaire
1	37.32	319	41.57
2	37.76	483	47.11
3	34.77	559	42.95
4	35.24	552	43.81
5	44.32	341	44.15
6	31.65	556	32.72
7	65.33	109	69.17

TABLE 2.1 – Tableau des résultats de la classification des 7 produits avec les arbres de décision.

Comme on le constate, ces résultats de classification sont très mauvais : six des sept expérimentations donnent un classifieur dont les résultats sont moins bons qu'une simple affectation à la classe majoritaire. Seul le classifieur pour le produit 5 dépasse de quelques points l'affectation à la classe majoritaire en donnant 44.32 % de bonne reconnaissance - pourcentage vraiment bas. De plus, les modèles appris sont complexes avec souvent entre 300 et 500 feuilles.

La conclusion que nous avons tirée de ces expérimentations est que les données que nous avons à notre disposition ne décrivent pas correctement le problème et ne permettent pas de construire un classifieur satisfaisant. Pour corriger cela nous avons cherché, sans succès, à enrichir nos données : les sources que nous avions à notre dispositions ne donnaient rien de plus et il n'était pas possible, pour des raisons de coûts, que l'entreprise acquière des données complémentaires. C'est pour toutes ces raisons que nous avons proposé de nous orienter vers une caractérisation des classes - le but étant d'extraire le plus de connaissances possibles de nos données.

2.1.3 Découpage des classes de prescripteurs

Dans le but de tirer le plus de connaissances possibles de notre jeu de données et de nos classes de prescription nous avons appliqué notre méthode de découpage automatique à notre base. Nous allons décrire, dans cette partie, nos expérimentations ainsi que les résultats que nous avons obtenus.

Etant donné que nous sommes en présence de données réelles avec des descripteurs très hétérogènes nous avons commencé par normaliser les données. Pour cela nous avons centré en retranchant à chaque valeur la moyenne de l'attribut. Puis nous avons divisé chaque valeur par l'écart type constaté du descripteur. Cette étape a été effectuée pour s'assurer qu'un attribut n'a pas plus d'importance qu'un autre dans le calcul des distances. La seconde étape a été de découvrir, pour trois valeurs de tolérance différentes (0, 5 et 10) et pour les sept jeux de données, les sous-classes présentes. La dernière étape a consisté à filtrer les jeux de données en ne conservant que les sous-classes qui contenaient plus de 10 exemples.

Le tableau 2.2 rassemble les résultats que nous avons obtenus : pour chaque produit (colonne 1) et pour chaque paramètre de tolérance (colonne 2) nous avons noté le nombre de sous-classes découvertes et, parmi celles-ci, combien dépassaient la taille de 10 exemples. La dernière colonne indique le pourcentage d'exemples conservés après filtrage des « petites » sous-classes.

Comme nous pouvions l'anticiper, on constate que plus le paramètre de tolérance augmente et plus le nombre de sous-classes découvertes diminue. Quel que soit le produit, on passe généralement de 1500 sous-classes pour une tolérance nulle, à moins de 100 pour une tolérance égale à 10. En effet, les exemples peuvent plus facilement se lier et les réseaux d'amis peuvent ainsi s'étendre plus loin.

Cependant, on remarque que, si le nombre global de sous-classes est divisé par 100, le nombre des « grosses » sous-classes, c'est-à-dire celles qui ont un nombre relativement élevé d'exemples, a tendance à être divisé seulement par 2, quand la tolérance varie entre

0 et 10. On remarque même qu'entre un réglage de tolérance à 0 et un autre à 5, le nombre de sous-catégories importantes ne croît que légèrement. Pourtant, le pourcentage d'individus conservés entre ces deux réglages augmente énormément en passant, en moyenne, de 10% à 80%. On constate donc que, pour une augmentation de la tolérance entre 0 et 5, il n'y a pas une forte émergence de nouvelles sous-classes masquées par du bruit, mais plutôt un accroissement en taille des sous-classes déjà identifiées. Ce phénomène s'explique par le fait que les classes sont mélangées : dans beaucoup de zones de l'espace des données plusieurs classes cohabitent et se chevauchent, empêchant ainsi l'émergence de nouvelles sous-classes.

Autre fait intéressant, si on regarde l'évolution globale du nombre de sous-classes significatives en fonction de la tolérance, on constate que celui-ci a tendance à légèrement croître quand la tolérance passe de 0 à 5 puis à décroître de 5 à 10. Il y a donc deux phases : la première est, comme nous l'avons dit plus haut, due à l'apparition de quelques nouvelles sous-classes, et la seconde vient de la fusion de sous-catégories : les réseaux d'amis se sont suffisamment étendus pour se lier entre eux. Il n'y a que le produit 7, qui contrairement à tous les autres, ne voit pas son nombre de sous-classes diminuer lorsque la tolérance passe de 5 à 10. Cette différence est dûe au très fort déséquilibre des classes au sein de ce jeu de données : la classe majoritaire est éparpillée dans tout l'espace des données et les réseaux d'amis ne se sont pas encore rejoints. On voit, en effet, que le nombre d'exemples conservés est de 84% pour une tolérance de 10 alors que pour les autres produits ce nombre est plutôt autour de 90%, voir 98% pour le produit 1.

Dernière constatation : au sein de chaque classe, des comportements différents ont été identifiés. Par exemple, le produit 3, pour une tolérance de 5, compte 13 sous-classes qui rassemblent 77% du nombre total d'exemples. On peut donc dire qu'il y a 13 zones de notre espace où une des classes est plus représentée que les autres puisque notre méthode y découvre un groupe compact. C'est une information capitale dans le processus de caractérisation du jeu de données. Cependant, nous ne sommes pas en mesure d'expliquer pourquoi ces trois classes recouvrent 13 aspects différents.

Ainsi, en plus de correctement souligner les différents comportements, cette expérimentation met en évidence un des autres atouts de notre approche qui est de fournir une indication sur la propension des classes à se chevaucher. En effet, l'étude de l'évolution du nombre de sous-classes significatives en fonction des variations du paramètre de tolérance permet de se faire une représentation du degré de mélange entre classes. Cependant, nous sommes conscients que la notion de classe significative repose, pour le moment, sur un

critère de taille et nécessiterait une formalisation plus poussée.

2.1.4 Utilisation du découpage pour l'apprentissage

Comme nous l'avons souligné, notre méthode permet de savoir à quel point les classes d'un jeu de données se chevauchent. Cela tient au fait que notre méthode segmente très largement les zones de chevauchement et laisse quasi-intactes les zones où une classe est très majoritaire. Dans cette section, nous allons utiliser cette information pour faire de la sélection d'exemples. Nous montrerons qu'il est possible, par le biais de notre approche, d'isoler les zones dites faciles à apprendre de celles qui demandent plus d'efforts.

Nous avons choisi de nous concentrer sur le produit numéro 3 de nos jeux de données. C'est en effet celui qui donne les scores d'apprentissage les moins mauvais du lot et qui présente une répartition des classes relativement équilibrée. Sur cette base, nous avons appliqué notre méthode de segmentation de données en faisant varier le paramètre de tolérance de 0 à 5 puis en le faisant passer à 10. Sur ces sept *runs* nous avons isolé les exemples appartenant aux sous-classes dont la taille est strictement supérieure à 10 exemples - les sous-classes significatives. Sur chacun de ces jeux de données sélectionnées, nous avons utilisé le logiciel Weka pour construire des arbres de décision en suivant le même protocole que dans la partie 2.1.2 (page 90), c'est-à-dire que nous avons découpé 30 fois de suite les données en deux tiers de construction du modèle et un tiers de validation puis fait la moyenne des taux de bonne classification. L'indice de confiance est de 0.25 pour l'élagage et on souhaite au moins 2 exemples par feuille. Le tableau 2.3 (page 97) regroupe les résultats que nous avons obtenus. La première colonne est la valeur du paramètre de tolérance, la seconde est le nombre de sous-classes significatives découvertes par notre approche pour cette valeur de tolérance. Dans la troisième colonne nous indiquons le nombre d'exemples que contiennent les sous-classes significatives avec entre parenthèse le pourcentage du total que ce nombre représente. Ce sont les exemples qui sont gardés pour apprendre les arbres de décision. Le taux de bien classés est donné dans la colonne quatre. Enfin, nous donnons à titre indicatif le nombre moyen de feuilles des arbres construits.

Comme pour l'expérience précédente on constate qu'il y a deux phases dans l'évolution du nombre de sous-classes significatives en fonction de l'augmentation de la tolérance. Pour une tolérance de 0 à 2, le nombre de sous-catégories importantes augmente. Puis, au-delà de 2, on assiste à une diminution du nombre de sous-classes significatives. Le phénomène est toujours le même : émergence de sous-classes, puis fusion de celles-ci les unes avec les autres.

De plus, ce qui est intéressant ici, c'est que les taux de bonnes classifications sont très hauts pour une tolérance faible et diminuent quand celle-ci augmente. Il est possible, par exemple, de classer avec une erreur de 30% la moitié du jeu de données - comme on le voit pour une tolérance de 2. Encore mieux : 7 % des exemples sont classés quasi parfaitement avec un arbre n'ayant que 5 feuilles ! On peut donc, grâce à la tolérance, faire de la sélection d'individus pour faciliter l'apprentissage d'un modèle.

On peut critiquer ces résultats en arguant du fait qu'en tirant au hasard des exemples d'un jeu de données et en construisant un modèle sur ce sous-jeu on obtient aussi de meilleurs résultats. En effet, il y a de fortes chances que les exemples soient éloignés les uns des autres et qu'un classifieur aura donc plus de facilité à les classer. Nous ne nions pas que l'éloignement facilite l'apprentissage. Mais, dans notre cas, les exemples ne sont pas éloignés les uns des autres. Au contraire, des exemples d'une même sous-classe seront, par construction, très proches les uns des autres. En tirant au hasard des groupes d'individus proches, il y a une grande probabilité qu'ils soient de classes différentes ce qui donnerait des résultats bien plus mauvais en terme de classification et de complexité de modèle que ce que nous obtenons.

On constate, par conséquent, que notre méthode permet de sélectionner efficacement les exemples représentatifs pour constituer des sous-jeux de données qui vont être beaucoup plus simples à apprendre. Cela représente un gain de temps et de complexité. En effet, il y a moins d'exemples à traiter et, étant donnée leur localisation dans l'espace des données, les modèles sont plus simples à construire. En somme, notre méthode isole les zones où une classe est très majoritaire. Or ces zones sont très simples à caractériser et à apprendre ; ce sont les zones de chevauchement de classes qui posent problème aux classifieurs. La figure 2.1 illustre ce principe : A gauche trois classes sont représentées par des surfaces. A droite on a représenté les mêmes données sans les zones de chevauchement - seules restent les zones simple à caractériser. Notre méthode joue ce rôle de filtre.

2.1.5 Synthèse

Dans cette section nous nous sommes intéressés à des données réelles issues du marketing pharmaceutique. Le but était d'essayer de caractériser la prescription de médecins pour sept médicaments. Nous avons commencé par appliquer les méthodes usuelles d'apprentissage, comme les arbres de décision, mais les résultats se sont révélés très décevants. Nous avons donc décidé d'utiliser notre approche de détection de sous-catégories pour mettre en avant les différents comportements présents dans les jeux de données. De plus, nous nous sommes concentrés sur un produit en particulier et nous avons montré qu'avec notre méthode on pouvait sélectionner les exemples pertinents et ainsi minimiser la com-

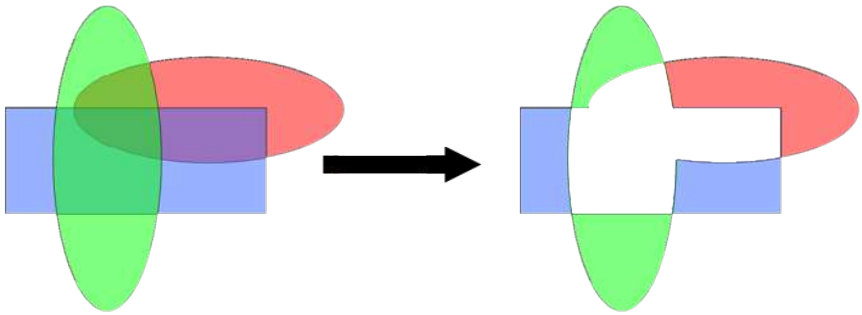

FIGURE 2.1 – Illustration de la suppression des zones de chevauchement dures à prédire.

plexité des modèles tout en améliorant les résultats.

Cependant il faut noter que bien que nos objectifs ait été atteints, il ne nous a pas été possible d'aller plus loin, faute d'expert pour analyser les résultats ou de nouvelles données pour enrichir les connaissances.

Produit	Paramètre de tolérance	Nombre de sous-classes	Sous-classes contenant plus de 10 exemples	Pourcentage conservé
1	0	1457	12	7 %
	5	145	12	89 %
	10	24	6	98 %
2	0	1493	15	13.5 %
	5	268	17	82 %
	10	91	6	94 %
3	0	1574	13	8 %
	5	319	13	77.5 %
	10	128	7	90 %
4	0	1572	13	10 %
	5	359	15	75.5 %
	10	124	10	90 %
5	0	1327	13	18 %
	5	169	9	87 %
	10	51	5	96 %
6	0	1634	5	2 %
	5	228	10	85 %
	10	101	6	93 %
7	0	886	8	53 %
	5	359	7	72.7 %
	10	206	14	84 %

TABLE 2.2 – Tableau des résultats de découpage en sous-classes des 7 produits.

Paramètre de tolérance	Nombre de sous-classes significatives	Cardinalité (% du total)	Taux de bonnes classifications	Nombre moyen de feuilles
0	10	166 (7,14%)	95,68	5
1	21	684 (29,41%)	81,85	26
2	21	1157 (49,74%)	71,10	60
3	17	1449 (62,30%)	59,56	83
4	14	1635 (70,29%)	53,94	105
5	13	1806 (77,64%)	49,52	142
10	6	2090 (89,85%)	42,80	146

TABLE 2.3 – Tableau des résultats d'apprentissage d'arbres de décision sur des jeux d'exemples sélectionnés pour le produit 3.

2.2 Prototypes sur données réelles

Nous avons montré que notre approche offrait de solides atouts pour la caractérisation de données artificielles et réelles. Nous allons poursuivre dans cette voie et nous concentrer sur l'utilisation conjointe des sous-classes et des méthodes à base de typicalité dans le but de construire des prototypes sur données réelles. C'est en effet une étape supplémentaire pour obtenir des résumés de données performants.

Cette partie est divisée en deux. Dans un premier temps nous montrons que les prototypes sur sous-classes que nous avons vu dans la partie 1.3 peuvent être appliqués sur des données réelles pour améliorer le pouvoir de généralisation des prototypes. Puis, dans un second temps, nous introduirons une variante de cette méthode en construisant des prototypes de valeurs d'attributs.

2.2.1 Création de prototypes

Nous avons construit des prototypes sur plusieurs jeux de données de l'UCI (UCI, 2009). Pour chacune de ces bases le même protocole a été suivi : la première étape consiste à normaliser les données pour s'assurer qu'aucun attribut ne va prendre le pas sur les autres. Ensuite, notre méthode de découverte de sous-comportements est utilisée sur les données. Le paramètre de tolérance est réglé à 0 car nous souhaitons, dans ces expériences, nous focaliser sur les sous-groupes très compacts. Pour chacune des sous-classes significatives, nous calculons les degrés de typicalité moyens. Enfin, nous comparons nos résultats avec la typicalité moyenne que l'on peut trouver sans utiliser notre méthode. Nous nous sommes intéressés à trois bases de données de l'UCI. Les résultats sont rassemblés dans le tableau 2.4 (page 99.

La première base de données est la base *Wine*. Elle est constituée de 178 exemples pour 12 attributs numériques et 3 classes. En regardant les résultats de décomposition, on peut constater que notre méthode n'a pas découvert plusieurs comportements par classes. En effet, les degrés de typicalité obtenus sans segmentation sont relativement bons pour les classes 1 et 3, et assez passables pour la classe 2. Les classes sont donc déjà relativement compactes et il n'y a pas lieu de découper. On constate également qu'en appliquant notre méthode les scores de typicalité moyens sont légèrement augmentés. Cependant cette amélioration est à mettre sur le compte du filtrage.

Le second jeu de données concerne des images provenant de satellites (*Statlog (Landsat Satellite)*). Il se compose de 2000 exemples décrits par 36 attributs numériques et répartis en 6 classes différentes. En comparant les degrés de typicalité obtenus avec et sans notre méthode, on constate qu'en moyenne nous améliorons sensiblement les résultats. Par

Bases de données	Classes	Typicalités moyenne sans segmentation	Typicalités moyenne avec segmentation de classes
Wine	1	0.7783	0.7869
	2	0.6033	0.6308
	3	0.7628	0.7221
Statlog	1	0.7356	0.7136
	2	0.8016	0.8108
	3	0.8871	0.9227
	4	0.8198	0.8320 et 0.9632
	5	0.7060	0.7223
	6	0.8392	0.8674
Glass	1	0.7505	0.8373, 0.9501, 0.9275
	2	0.5406	0.89, 0.94, 0.97, 0.62, 0.77
	3	0.7003	
	4	0.3796	0.7413
	5	0.5731	0.8275
	6	0.6694	0.7322

TABLE 2.4 – Tableau des degrés de typicalité moyens par classes sans segmentation et avec une segmentation par réseaux d'amis pour trois jeux de données de l'UCI.

exemple, dans la classe 4, qui obtenait un score satisfaisant de 0.82, deux comportements ont été mis en avant avec des degrés de 0.83 et 0.96 ce qui augmente encore la qualité des prototypes que l'on peut obtenir.

Le dernier jeu de données s'intéresse à différents types de verres (base *Glass Identification*). 214 exemples décrits par 6 attributs numériques sont répartis en 6 classes. Pour 5 classes sur 6 les degrés de typicalité ont été améliorés. Les classes 1 et 2 ont été segmentées en respectivement 3 et 6 sous-classes et dans chaque cas la typicalité moyenne de la sous-classe n'est pas inférieure à 0.77 et peut atteindre 0.97. Cependant, il faut noter que la classe 3 a été segmentée en 13 sous-classes non significatives et n'apparaît donc plus dans les résultats finaux. Cela s'explique tout d'abord par le fait que la classe n'est pas grande (17 individus sur 214) mais c'est surtout la répartition des exemples dans l'espace des données qui conduit à cette situation : le fait que la propagation des réseaux d'amis soit très restreinte, indique que cette classe est mélangée avec les autres classes.

Les deux seules fois où la typicalité moyenne obtenue avec notre méthode est inférieure à celle obtenue sans segmentation, i.e. la classe 3 de la base *Wine* et la classe 1 de *Statlog*,

s'expliquent par le rejet d'individus des autres classes faisant ainsi baisser la dissimilarité externe et donc la typicalité. Il est possible que ces individus soient éloignés de toutes les classes.

2.2.2 Prototypes de valeurs

Nous allons à présent montrer une utilisation différente de notre approche. En effet, précédemment, nous avons construit des prototypes de classe en utilisant plusieurs exemples du jeu de données qui les résumaient correctement. A présent, nous nous intéressons aux attributs pris séparément. Le but est de donner des valeurs prototypiques d'attributs. Ce travail a été réalisé dans le cadre d'un article (Forest et al., 2006a) puis enrichie dans une seconde publication (Forest et al., 2006b).

Pour la mise en pratique de notre algorithme, nous avons choisi de travailler sur la base de données de classifications des verres (*Glass Identification*) provenant de l'UCI data repository (UCI, 2009). 214 exemples de six types de verres différents (les classes) sont décrits par 9 attributs numériques. Dans cette étude nous nous sommes concentrés sur le premier attribut (l'indice de réfraction) car c'est l'attribut pour lequel la détection de valeurs prototypiques sans segmentation des classes donne les meilleurs résultats. La figure 2.2 montre les prototypes flous obtenus sans segmentation des classes.

Comme il est suggéré dans Lesot (2005b), nous considérons qu'une valeur est significative (i.e. typique) si son score de typicalité est au dessus d'un certain seuil, 0.7 dans notre cas.

Sans la segmentation des classes, on peut observer deux types de classes. Les scores de typicalité des prototypes des classes 1, 3 et 6 sont significatifs alors que ceux des trois autres (2, 4 et 5) restent sous le seuil. Ce résultat peut sembler non satisfaisant. Cependant, il est important de noter que la méthode classique a été créée pour un contexte particulier : celui de classes provenant d'un clustering ou de classes bien définies (i.e. des classes homogènes). La base de données de la figure 2.2 montre les limites de l'approche dite classique et met en avant le besoin d'une phase de prétraitement pour le cas de classes non-homogènes.

L'expérimentation met en évidence différents points :

Meilleure caractérisation

La figure 2.3 illustre le bénéfice de notre processus de segmentation. Dans cette représentation, toutes les sous-classes ne sont pas représentées : nous avons choisi de filtrer les

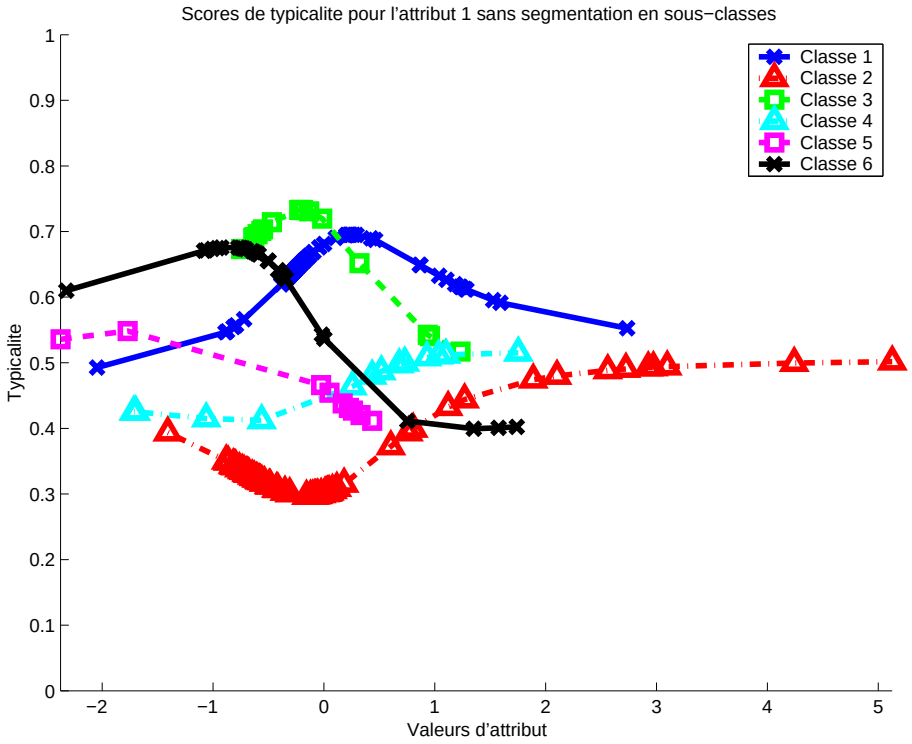

FIGURE 2.2 – Scores de typicalité en fonction des valeurs du premier attribut sans segmentation des classes.

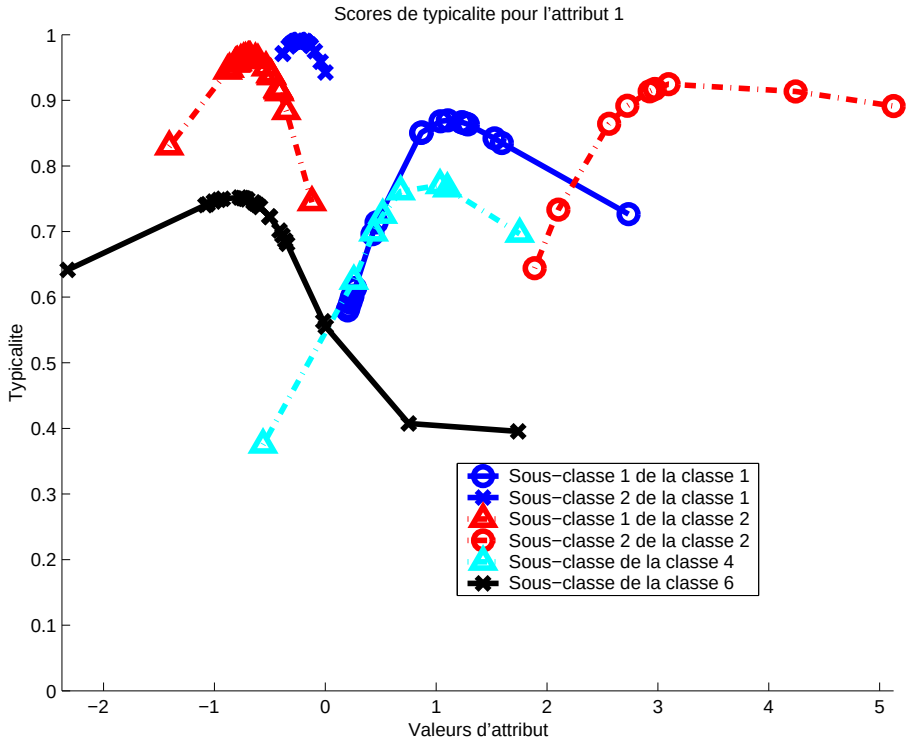

FIGURE 2.3 – Scores de typicalité pour le premier attribut de la base de données des verres avec la segmentation des classes et le filtrage des sous-classes (seuil à 3%).

sous-classes contenant moins de 3% du nombre total d'individus. Nous insistons sur le fait que les classes filtrées ne sont pas apparentes sur le graphe mais qu'elles sont cependant prises en compte lors du processus.

Les classes bleue (1) et rouge (2) sont ici mieux caractérisées : celles-ci ont chacune été segmentée en deux sous-classes majeures et les scores de typicalité de leurs prototypes sont hauts (autour de 0.9). A présent, ces classes ont deux plages de valeurs typiques. Il est important de remarquer que les sous-classes obtenues n'ont pas toutes la même représentativité compte tenu du nombre d'individus qu'elles contiennent. En effet, la classe bleue (1) donne ici plusieurs prototypes qui rassemblent 71% des individus alors que les prototypes de la classe rouge contiennent 48.5% du nombre d'individus de la classe de départ.

Stabilité

La classe cyan (4) et la classe noire (6) illustrent une autre situation intéressante. Pour ces deux classes la segmentation donne plusieurs sous-classes avec une seule au dessus du seuil des 3%. Cela signifie que les prototypes bien définis, obtenus sans la segmentation, sont correctement restitués par notre procédure de segmentation.

Effet de seuil

On peut noter que les classes rose (5) et verte (3) ont disparu dans la figure 2.3. Ces classes ont été si segmentées qu'aucune sous-classe ne dépasse le seuil. Il est évidemment possible de modifier le seuil.

La figure 2.4 montre qu'avec un seuil de 1% les classes rose (5) et verte (3) réapparaissent. Cependant, six nouveaux prototypes pour la classe rouge (2) émergent, ce qui rend la caractérisation plus difficile.

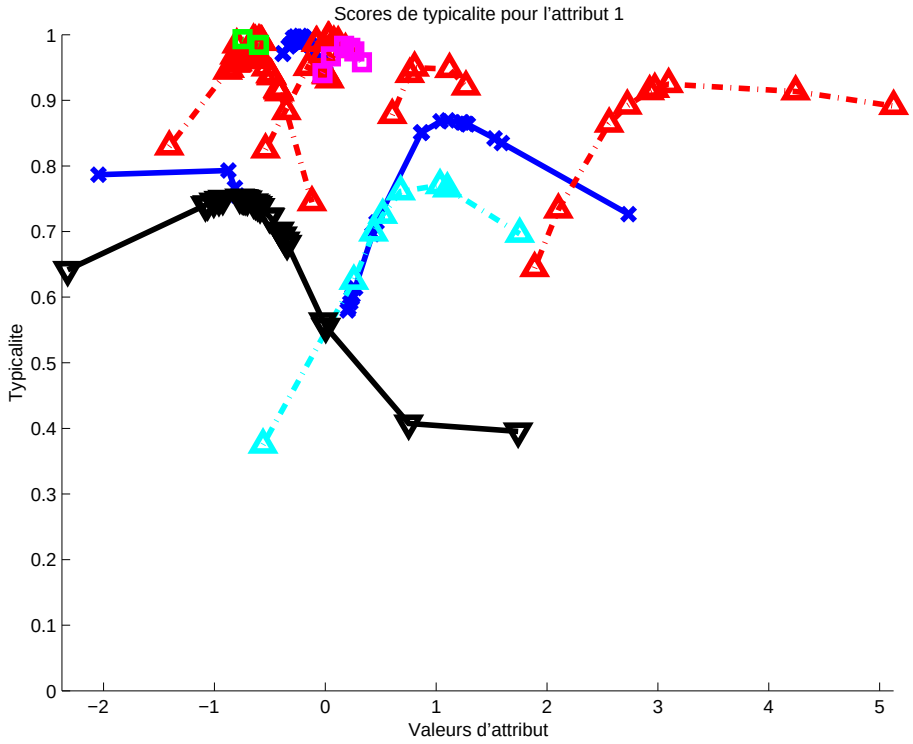

FIGURE 2.4 – Scores de typicalité pour le premier attribut de la base de données des verres avec la segmentation des classes et le filtrage des sous-classes (seuil à 1%).

2.3 Résumé d'une base d'images

Cette partie du chapitre est dédiée aux travaux que nous avons effectués sur le résumé d'information. Comme nous l'avons évoqué précédemment, les méthodes de découverte de prototypes ont pour vocation à mettre en avant les exemples qui donnent la meilleure représentation de leur catégorie. Ils peuvent donc être utilisés pour résumer la classe qui les contient. Nous avons utilisé notre méthode de calcul de prototypes pour fournir une résumé visuel d'une collection d'images. Nous commencerons par décrire les données. Puis, nous poursuivrons en expliquant le protocole de notre expérience. Nous exposerons ensuite les principaux résultats obtenus avant de conclure.

La base d'images sur laquelle nous avons centré cette étude est connue comme la base *Corel 1000* de l'université d'état de Pennsylvanie (Li & Wang, 2003). Elle se compose de 1000 images réparties uniformément en 10 thèmes : Afrique, plages, monuments, bus, dinosaures, éléphants, fleurs, chevaux, montagnes et nourriture. L'intérêt de cette base est que les thèmes sont très variés. En effet, on y trouve des photos de paysages, de monuments, de personnes, d'animaux, etc. De plus, certaines catégories sont composées d'images très similaires (comme les dinosaures ou les fleurs) alors que d'autres regroupent sous un thème assez abstrait des images très différentes (par exemple monuments ou plages). Dans l'annexe 3 page 141, le lecteur peut trouver un résumé de la base de données.

Pour ce qui est des données de description, comme notre méthode de découpage prend en entrée des données numériques et calcule des matrices de distances, pour être en mesure de traiter ces images il nous a fallu nous procurer des descriptions sous formes de vecteurs. Nous avons fait appel à nos collègues qui ont développé une méthode d'extraction d'information basée sur les couleurs de l'image. Le détail de cette méthode est exposé dans Marsala et Detyniecki (2006). En résumé, il s'agit de découper l'espace des couleurs HSV (Hue, Saturation and Value) et de compter combien de pixels il y a dans chaque « morceau » de l'espace. En pratique, nos images ont été décrites selon 24 descripteurs : la composante H a été divisée en 6, la composante S en 2 et la composante V en 2 également. Cela donne $6 \times 2 \times 2 = 24$. En outre, la classe d'une photographie est le thème auquelle elle appartient.

Ainsi, nous disposons pour chaque image d'un vecteur de description numérique et d'une classe. Nous pouvons donc appliquer notre méthode de découverte de sous-classes pour trouver les sous-catégories d'images. La seconde étape est la sélection des sous-classes significatives. En effet, nous considérons qu'une sous-classe est significative si elle repré-

sente au moins 4% de la taille de sa classe ; ce qui, dans notre cas, représente 4 exemples. Nous mettons donc de coté les exemples qui sont trop isolés de leur classe. La troisième étape est le calcul des degrés de typicalité des photographies. Comme dans les expériences précédentes, nous utilisons des mesures de ressemblances et de dissimilarité basées sur la distance euclidienne normalisée. Ces deux mesures sont agrégées par la somme symétrique. Enfin, pour résumer une catégorie nous proposons d'afficher, pour chaque sous-classe significative de celle-ci, l'image qui maximise le degré de typicalité.

Les résultats de notre processus de résumé d'information appliqué à la base *Corel 1000* se trouve sur la figure 2.5. Chaque carré est le prototype d'une classe - le nom de la classe, son numéro et sa taille sont indiqués dans le texte en haut. Chaque image contenue dans un carré est l'image la plus typique de sa sous-classe. En dessous des images, nous avons indiqué la typicalité maximum de la sous-catégorie ainsi que le nombre d'exemples qu'elle représente.

Sur la figure 2.5 (page 109), on constate que notre méthode a résumé les 1000 images en 10 prototypes. Dans les prototypes obtenus il n'y a jamais plus de 3 images, ce qui rend la lecture très simple et très intuitive. Nous avons choisi de représenter l'image de la plus-grande sous catégorie de la classe à gauche et légèrement plus grande. On peut également remarquer, qu'en moyenne, les scores de typicalité oscillent entre 0.7 et 0.8 ce qui est très bon et prouve que les images choisies représentent bien leur sous-catégorie. Si on compare les prototypes construits par notre méthode avec la base d'images, on se rend compte que les tendances mises en avant reflètent correctement celles qui existent dans la base [4].

En poussant un peu plus l'analyse on peut identifier trois familles de prototypes :

– La première famille que nous pouvons voir sur nos résultats est celle représentée pas les classes 6 et 7 (Fleurs et Eléphants). En effet, ces classes sont toutes deux représentées pas deux comportements. En outre, les deux sous-classes significatives regroupent un très grand nombre d'exemples (respectivement 86 et 70 sur 100). Cela signifie que la phase de filtrage n'a pas éliminé un trop grand nombre de sous-catégories et, par conséquent, que les classes n'en chevauchent pas d'autres de façon trop prononcée. Il est intéressant de noter que le fait de découvrir deux comportements distincts colle parfaitement avec la réalité des données. Par exemple la classe des fleurs comporte quasi exclusivement des roses dans les tons roses/rouges et d'autres dans les tons jaunes – deux caractéristiques très typiques que l'on retrouve

4. Nous rappelons que nous avons mis en annexe, page 141, les images de chaque classe de la base.

ici.

- Le deuxième type de prototype regroupe les classes qui n'ont qu'une sous-classe majoritaire. C'est le cas des classes 1, 4, 5, 8 et 10. La typicalité de ces prototypes est relativement bonne (en moyenne 0.75 sauf pour la classe 8, les chevaux, qui est à 0.62). De plus, le nombre d'exemples que le prototype regroupe est très élevé et peut même atteindre 99 sur 100 pour la classe 5. Cela signifie que ces classes sont très compactes et ne se mélangent pas avec les autres. Or, si on regarde en détail les images de la base on remarque, qu'en effet, ces classes affichent des images très ressemblantes : tous les dinosaures sont dessinés sur un fond blanc ou beige proéminent, les images de chevaux les représentent quasiment tous dans un pré à l'herbe très verte, etc. Les informations restituées par nos prototypes sont en accord complet avec ce qu'on peut observer en regardant en détail la base.
- La troisième famille de prototypes regroupe les classes 2, 3 et 9 ; plages, monument et montagnes. Celles-ci sont restituées par deux ou trois comportements différents. Cependant, ce qui les caractérise surtout, c'est que les sous-classes significatives qui les composent sont relativement petites avec en moyenne un peu moins de 40 exemples sur 100. Cela signifie que notre méthode de découverte de sous-classes a fortement découpé ces données en très petites sous-classes. Ce phénomène s'explique par le type d'images présentes dans la classe : en effet, ce sont des classes présentant des images très variées. Par exemple, il y a beaucoup de plages différentes et les images se focalisent tantôt sur le ciel, tantôt sur le sable, créant ainsi une grande disparité. Même chose pour les montagnes qui peuvent être enneigées ou non. La plus grande non-homogénéité se rencontre dans la classe des monuments car celle-ci rassemble sous la même étiquette, des édifices très variés, photographiés de l'extérieur ou de l'intérieur. Toutes ces caractéristiques très diverses font que les exemples se retrouvent distants les uns les autres dans l'espace de description. Notre méthode se focalise exclusivement sur les caractéristiques propres à une classe. L'absence d'exemples, plus ou moins prononcée, est un indicateur important de la diversité de la classe.

Concentrons-nous à présent sur un autre indicateur très important : la typicalité. Il est intéressant de voir que la typicalité et la taille des sous-classes ne sont pas proportionnelles. En effet, on peut prendre en exemple les classes 5 et 8 (Dinosaures et Chevaux) – dans la première la taille est très élevée et la typicalité également (0.83) alors que dans la seconde, bien que la taille de la sous-classe significative soit très élevée, la typicalité n'est que de 0.62. Cette différence s'explique par le fait que la topologie des deux classes est différente. Dans les deux cas les exemples des classes sont proches, c'est-à-dire qu'ils

partagent tous des caractéristiques communes que les autres images de la base n'ont pas. Cependant l'organisation spatiale de la seconde classe fait baisser la typicalité. En somme, la seconde classe est moins compacte que la première.

En conclusion, nous avons présenté une méthode de résumé de base d'image qui s'appuie sur la découverte de sous-classes et le calcul des degrés de typicalité. Nous avons testé cette méthode sur la base *Corel 1000* et nous avons montré que nos prototypes restituaient correctement les tendances des classes même dans le cas où plusieurs étaient présentes au sein d'une classe. De plus notre méthode, par le biais des indicateurs de taille et de typicalité, permet d'enrichir la caractérisation des classes en donnant des informations sur le degré de mélange d'une classe avec les autres, ainsi que sur la topologie des classes. Enfin, nous souhaitons rappeler que nos descripteurs s'appuient sur les couleurs des images. Il n'y a donc aucune sémantique dans notre analyse.

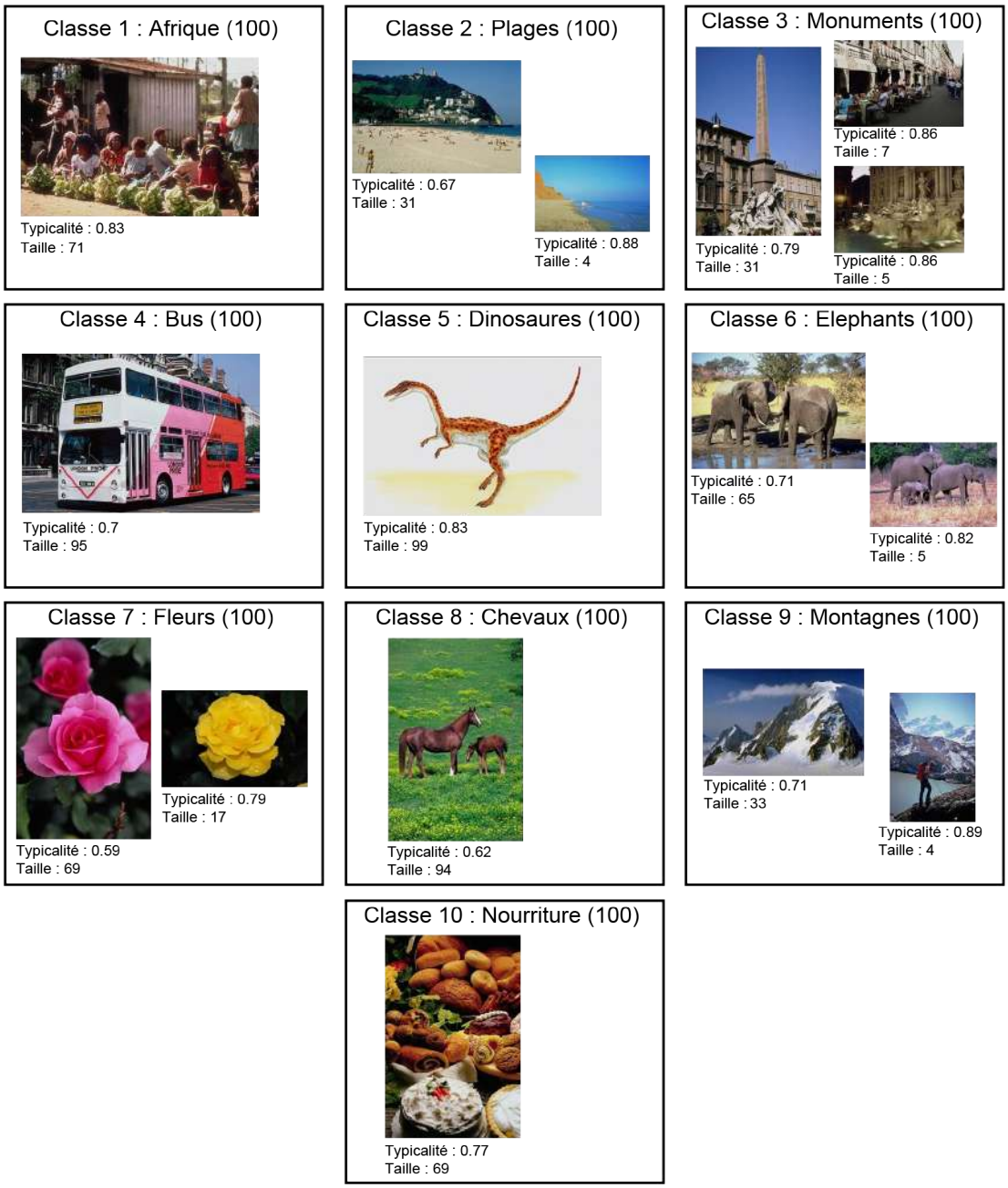

FIGURE 2.5 – Résumé par le biais de prototypes de la base Corel comportant 1000 images réparties en 10 classes.

Chapitre 3

Comparaisons de découpages

Dans ce chapitre nous nous intéressons à une autre méthode de découpage de l'espace : le clustering. Notre objectif est de comparer notre approche de segmentation avec celle utilisée dans les articles que nous avons présentés dans la partie 3.3.

Nous commençons notre étude par un rappel de l'algorithme de clustering utilisé. Puis, nous comparons les prototypes obtenus avec notre méthode et ceux obtenus avec les k-moyennes floues sur un jeu de données artificielles.

3.1 Les k-moyennes floues

Pour valider notre méthode, nous l'avons comparée aux k-moyennes floues (Dunn, 1973; Bezdek, 1981). Cet algorithme est une extension des k-moyennes (méthode de partitionnement de type « nuées dynamiques ») : l'affectation d'une donnée à un cluster n'est pas binaire comme pour les k-moyennes, mais représentée par un degré d'appartenance. L'algorithme est le suivant :

1. Initialiser aléatoirement k centres, w_r, $r = 1, \ldots, k$.
2. Itérer jusqu'à stabilisation
   (a) Calculer les degrés d'appartenance u_{ir} de la donnée i au cluster r.
   $$u_{ir} = \frac{1}{\sum_{s=1}^{k} (\frac{\|x_i - w_r\|}{\|x_i - w_s\|})^{2/(m-1)}}$$
   (b) Mettre à jour les positions des centres :
   $$w_r = \frac{\sum_{i=1}^{n} u_{ir}^m \cdot x_i}{\sum_{i=1}^{n} u_{ir}^m}$$

Cet algorithme a pour paramètre le degré de flou noté m. Sa valeur est traditionnellement de 2.

3.2 Comparaison sur des données artificielles

Dans cette expérimentation, nous travaillons sur un jeu de données artificiel composé de trois classes et décrit par deux attributs. La figure 3.1 en donne une représentation. On constate que les classes 1 et 3 sont non-homogènes et constituées chacune de 4 groupes.

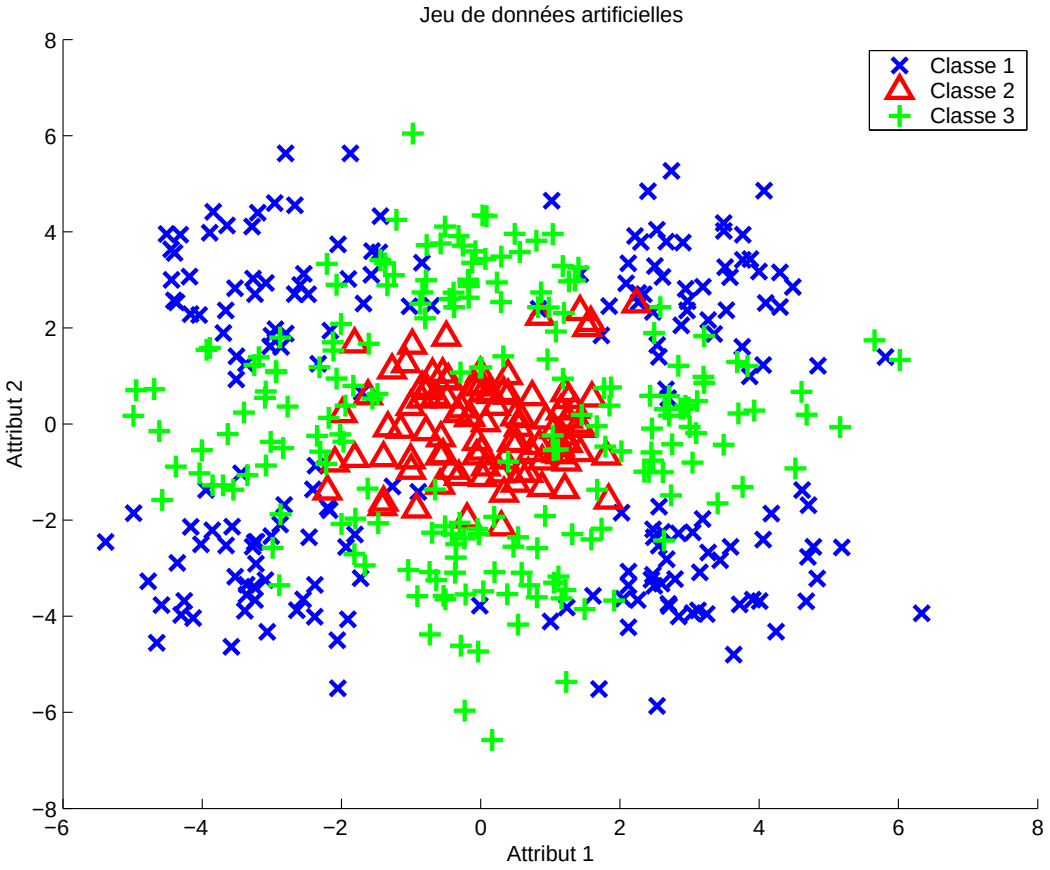

FIGURE 3.1 – Base de données artificielles partitionnée en 3 classes dont deux sont hétérogènes.

Pour garantir une comparaison impartiale, nous avons donné en paramètre aux k-moyennes floues le nombre de classes trouvé par notre méthode. De plus, l'algorithme a été appliqué à chaque classe séparément et non pas globalement. De cette manière, nous voulons montrer la cohérence de nos résultats par rapport à ceux fournis par les k-moyennes floues. Ainsi, on a appliqué l'algorithme avec $k = 4$ sur les classes 1 et 3.

La figure 3.2 (page 114) montrent les exemples qui maximisent la typicalité : en haut, ceux obtenus par notre méthode. En bas, ceux obtenus avec les k-moyennes floues. On remarque une similarité entre ces résultats et ceux obtenus par notre approche : les différentes sous-classes trouvées par les deux méthodes sont proches lorsque l'on compare les

exemples les plus typiques de chaque sous-classe.

Lorsque l'on calcule les degrés de typicalité des individus pour chaque sous-classe (table 3.1), on constate tout d'abord qu'il y a une amélioration des degrés de typicalité lorsque l'on segmente au préalable les classes, que ce soit avec notre méthode ou avec celle des k-moyennes floues. Ensuite, notre méthode donne de meilleurs résultats en termes de degrés de typicalité moyens.

Classes	Degrés de typicalité moyens		
	sans segmentation	segmentation par réseaux d'amis filtrés	segmentation par k-moyennes floues
Classe 1	0.3827	4 sous-groupes : 0.9448 0.9352 0.9304 0.9211	4 sous-groupes : 0.8845 0.8968 0.9018 0.8968
Classe 2	0.8349	0.8666	0.8084
Classe 3	0.5402	4 sous-groupes : 0.9473 0.8958 0.9122 0.9220	4 sous-groupes : 0.8674 0.8569 0.8614 0.8828

TABLE 3.1 – Tableau des degrés de typicalité moyens par classes sans segmentation, avec une segmentation par réseaux d'amis et une segmentation par k-moyennes floues.

3.3 Synthèse

On constate donc que notre approche donne de meilleurs résultats qu'une segmentation basée sur les k-moyennes floues. En effet, sur notre jeu de données, les prototypes trouvés sont comparables mais les degrés de typicalité calculés sont supérieurs avec notre algorithme.

De plus, le paramètre k de la méthode de clustering a été correctement réglé, ce qui n'est pas envisageable avec des données réelles.

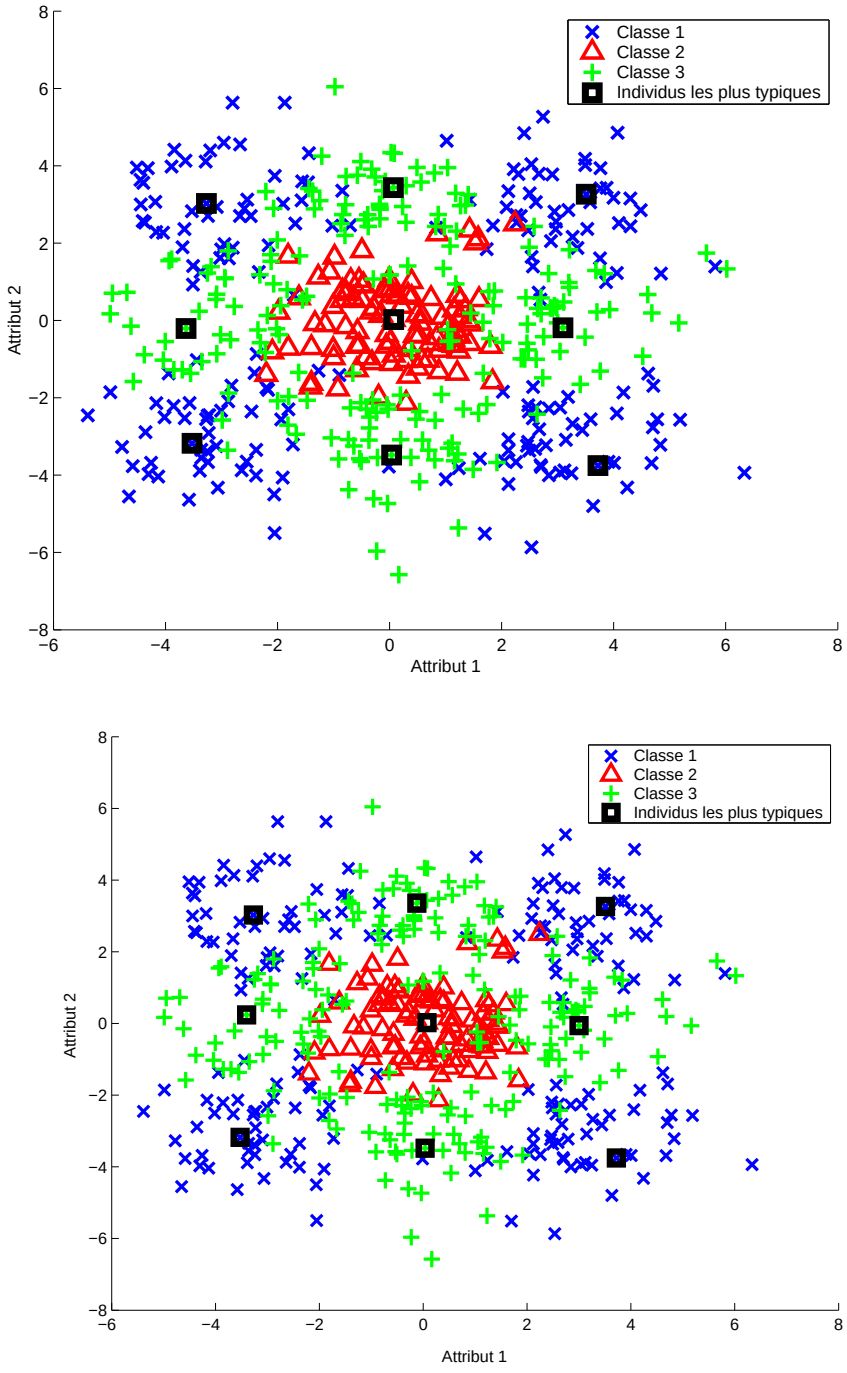

FIGURE 3.2 – En haut : exemples les plus typiques pour chaque sous-classe découverte par notre méthode. En bas : exemples maximisant la typicalité après avoir segmenté les classes séparément grâce aux k-moyennes floues

Conclusions

Notre thèse se focalise sur les méthodes de caractérisation de données dans un cadre supervisé. Nous avons fait un état de l'art des approches qui permettent de tirer le maximum d'information des classes ; que ce soit par le biais de résumés linguistiques ou le calcul de prototypes.

De plus, nous avons montré que ces méthodes ne donnaient pas de bons résultats dans le cas de classes non-homogènes car celles-ci sont composées de plusieurs comportements qui ne peuvent pas être résumés ou représentés à travers une seule tendance. Pour être correctement prises en compte et donc correctement caractérisées, ces classes nécessitent une décomposition en sous-classes.

Ainsi, un état de l'art des méthodes existantes de décomposition de classes a été réalisé. Grâce à celui-ci, nous avons vu que deux types d'approches cohabitaient : tout d'abord les méthodes de classification par hyperplans ou hypersphères, ensuite les méthodes faisant un clustering des classes. Dans les deux cas, nous avons expliqué que ces approches ne permettent pas une caractérisation optimale car elles sont soit concentrées sur une tâche de classification, et donc de généralisation, qui n'est pas compatible avec la caractérisation ; soit elles s'appuient sur un algorithme de classification non supervisée, ce qui ne permet pas d'appréhender correctement les dynamiques entre les classes et les zones de chevauchement interclasses.

Nous avons donc proposé une nouvelle méthode originale de décomposition de classes en sous-classes qui permet une caractérisation des classes en tenant compte des dynamiques interclasses. Notre méthode recouvre l'espace des données avec des hypersphères. Puis, elle tisse des liens entre les exemples qui sont contenus dans les sphères pour construire des *graphes d'amis*. Enfin, en parcourant les graphes obtenus, elle fournit, pour chaque exemple, une affectation à une sous-classe. Chaque sous-classe ainsi mise en évidence représente un comportement de la classe. Notre méthode a un paramètre, la tolérance, qui lui donne une résistance au bruit.

Nos travaux ont servi à caractériser les données de l'entreprise ARVEM en mettant

en avant la non-homogénéité des profils de prescription chez des médecins généralistes. Un webservice de construction de sous-classes a été mis en place et proposé à l'entreprise pour lui permettre le traitement de ses données futures.

Nous avons validé notre approche sur des données artificielles car celles-ci permettent une visualisation des résultats. Ainsi, notre méthode détecte efficacement les différents comportements tout en faisant preuve de parcimonie.

Puis, nous avons proposé une extension des travaux de Lesot (2005b) en rendant possible la construction de prototypes flous d'attributs sur des données contenant des classes non-homogènes. Nous avons montré qu'en présence d'un découpage en sous-classes, le calcul des degrés de typicalité pouvait être modifié pour prendre en compte les sous-catégories.

Grâce à notre travail sur les données de médecins généralistes, nous nous sommes rendu compte que notre méthode permettait de filtrer les exemples difficiles à apprendre par un classifieur. En effet, notre algorithme détecte les exemples qui sont dans les zones de chevauchement de classes. Ces zones sont ambigües et posent des problèmes aux méthodes d'apprentissage. Ainsi, en filtrant ces exemples il devient possible de construire des jeux de données moins précis mais qui peuvent être modélisés beaucoup plus simplement fournissant donc un gain de temps et de ressources.

Notre approche permet la détection efficace de sous-classes. En nous appuyant sur ce principe, nous avons proposé une nouvelle méthode de résumé d'information qui définit le prototype d'une classe comme l'ensemble des exemples les plus typiques de chaque sous-classe. Cette approche a été testée et validée sur une base d'images.

Enfin, nous avons comparé notre découpage des classes avec celui effectué par un algorithme de clustering. Notre méthode est plus performante car les prototypes obtenus sont de meilleure qualité et il n'y a pas besoin de spécifier le nombre de sous-classes recherchées.

En conclusion, nos travaux de thèse ont apporté un nouvel outil au domaine de la fouille de données. Celui-ci permet de construire des prototypes bien plus riches. L'originalité de notre approche se trouve dans la façon de prendre en compte les classes non-homogènes et les dynamiques interclasses.

Perspectives

Il y a un grand nombre de pistes de recherches que nous aimerions suivre. Nous les avons classées par ordre d'importance :

1 Sous-classes non séparées

Le défaut principal de notre méthode est qu'une classe est segmentée en sous-classes si et seulement si les sous-groupes qui la composent sont séparés les uns des autres par des individus d'autres classes. En d'autres termes, s'il n'y a que du "vide" entre deux sous-groupes, notre méthode ne sera pas en mesure de les identifier correctement.

La figure 1 illustre cette situation : la classe des croix est bien constituée de deux sous-classes que l'utilisateur n'a aucun mal à distinguer. Cependant, ces deux groupes ne sont pas détectés par notre approche. C'est un défaut que nous aimerions corriger dans le futur.

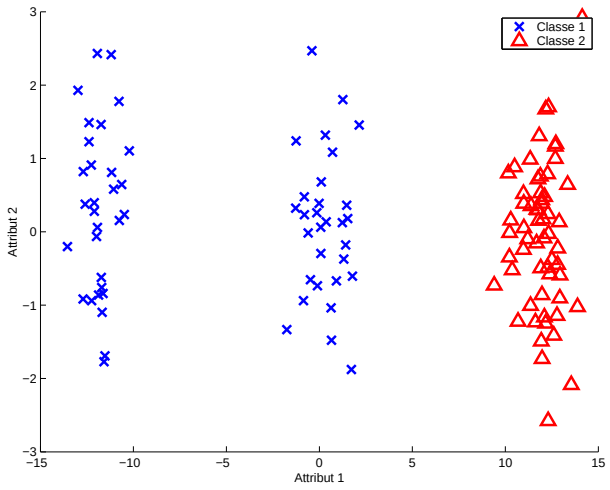

FIGURE 1 – Illustration du phénomène de non détection de sous-classes : si deux sous-groupes d'une classe ne sont pas séparés par un exemple d'une autre classe, alors notre méthode ne les détecte pas. C'est le cas de la classe bleue à gauche sur cette figure.

Nous avons déjà trouvé une heuristique pour y parvenir. Cependant, les expérimentations qui permettraient de la tester et de la valider ne sont pas terminées au moment où nous écrivons ce manuscrit. Nous pouvons tout de même expliquer succinctement cette amélioration.

La première étape consiste à *trouver les sous-classes qui n'ont pas été segmentées* alors qu'elles le devraient. Nous proposons pour cela, de calculer les degrés de typicalité en fixant un seuil minimal en dessous duquel on considère que la sous-classe n'est pas assez découpée.

La seconde étape est *la segmentation au « bon endroit »*. Nous allons pour cela utiliser le graphe d'amis du jeu de données. Nous rappelons que, dans ce graphe, les nœuds sont les exemples et les arêtes (non orientées) sont les liens d'amitié et pondérées par la distance entre les deux exemples. L'idée de notre amélioration est de construire l'arbre couvrant minimal (l'ACM) de ce graphe. C'est l'arbre qui minimise le nombre et le poids total des arêtes tout en gardant intacte la structure du graphe. Ainsi deux exemples qui étaient reliés, le restent après application de l'algorithme. Il existe plusieurs méthodes pour construire ce type de structure ; nous avons choisi d'utiliser l'algorithme de Kruskal (1956) pour sa simplicité.

La minimisation des arêtes du graphe nous garantit que les deux sous-classes ne seront reliées que par une unique arête. De plus, toujours par construction, celle-ci aura le poids le plus important du graphe qui constitue les groupes. En supprimant cette arête on identifie alors correctement les sous-classes.

La figure 2 illustre ce procédé. L'image en haut montre les graphes d'amitié entre les exemples de notre jeu de données. Puis, en appliquant l'heuristique de Kruskal sur ce graphe on obtient l'arbre couvrant montré sur l'image centrale. En identifiant sur celui-ci l'arête la plus grande et en la « coupant » (c'est-à-dire en éditant notre matrice d'adjacence) on constitue alors deux groupes – les groupes correspondant aux sous-classes recherchées.

On peut tout de même critiquer cette amélioration de la méthode car elle est itérative : tant que le seuil de typicalité (ou tout autre évaluation) n'est pas atteint par toutes les sous-classes il faut recalculer les degrés de typicalité, ce qui est très coûteux.

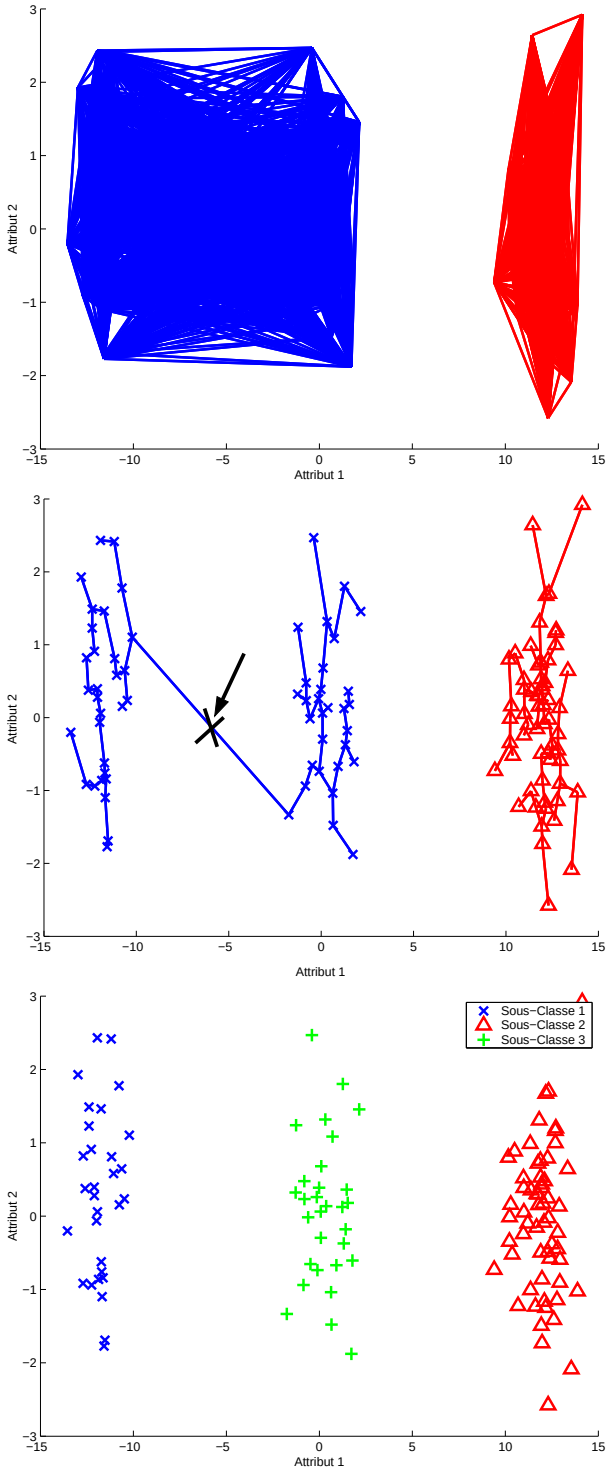

FIGURE 2 – En partant du graphes de liens d'amitié entre les exemples et en appliquant un algorithme d'arbre couvrant minimal on peut détecter l'arête la plus longue. En la supprimant on détecte alors correctement les sous-classes.

2 Délimitation des sous-classes

Les sous-classes que nous construisons n'ont, pour le moment, aucune frontière réelle, aucune limite clairement définie. Nous donnons pour chaque exemple l'affectation à sa sous-catégorie. L'étape supplémentaire que nous souhaiterions franchir est la construction automatique de frontière aux sous-classes.

Si nous disposions de telles limites nous pourrions envisager d'utiliser notre méthode dans le but de classifier des données non étiquetées. Celui-ci serait d'un genre nouveau car il donnerait l'appartenance d'un nouvel exemple à une sous-classe, ce qui est une information encore plus riche.

Pour y parvenir nous avons identifié une piste qui nous semble très intéressante : il existe un lien fort entre un exemple x d'une classe c et l'exemple y d'une autre classe qui a servi de limite à l'expansion de la sphère d'influence de x. On peut considérer que y est l'exemple limite de x. En utilisant, pour chaque exemple de la sous-classe, l'information d'exemple limite on peut esquisser des hyper-plans séparateurs et fermer les sous-classes.

Enfin, étant donné que la fermeture des sous-classes ne peuplerait que les zones denses de l'espace des données, notre classifieur serait aussi en mesure de coller une étiquette « indéterminé » sur les nouveaux exemples en dehors des limites des différentes sous-classes.

On peut noter, qu'en plus de cette piste, nous voudrions également faire un état de l'art des méthodes de construction d'enveloppe convexe de points qui sont, par exemple, utilisées en infographie et qui, à notre avis ont un fort potentiel dans notre cas.

3 Amélioration des forêts de décision

Dans notre travail nous avons montré dans la partie 2.1.4 que notre méthode permettait de filtrer plus ou moins fortement les exemples difficiles à apprendre, car appartenant à des zones de chevauchement de plusieurs classes. En poursuivant dans ce sens, nous aimerions utiliser cette propriété pour la construction de forêts d'arbres de décision.

Dans ce domaine, il est courant d'utiliser des méthodes stochastiques pour tirer les exemples qui serviront à apprendre les arbres de décision. Nous pensons que notre approche pourrait se substituer au tirage aléatoire pour permettre de se focaliser sur les exemples difficiles à apprendre. Il y aurait donc une minorité d'arbres assez simples se concentrant sur les zones où la décision est simple à prendre et une majorité d'arbres de la forêt consacrés aux zones problématiques et demandant plus de travail. Cela donnerait des classifieurs plus robustes et plus performants.

4 Clustering par exemples fantômes

Nous nous sommes demandé si notre méthode ne pouvait pas servir pour des données non étiquetées afin de constituer des groupes, à l'instar des algorithmes de clustering dont nous avons parlé précédemment. En investiguant, nous avons identifié une piste qui nous semble intéressante et qui mériterait d'être développée dans l'avenir.

En effet, comme nous l'avons expliqué dans ce manuscrit, nos sous-classes apparaissent du fait de la séparation des autres classes. Or, dans un cadre non supervisé il n'y a aucune contrainte car pas de classe. Notre proposition est donc de la créer artificiellement. Pour cela nous créons une classe d'exemples « fantômes » qui vont venir séparer les groupes existants. Il suffit alors d'appliquer notre méthode sur ce nouveau jeu de données constitué de deux classes et de garder les sous-classes qui nous intéressent.

Pour le moment, nos réflexions nous poussent à proposer une répartition uniforme de ces exemples « fantômes » selon une grille. Mais on peut imaginer d'autres types de dispersion qui utiliseraient le hasard par exemple.

La figure 3 donne une représentation de cette perspective de notre travail. Dans l'image du haut on voit le jeu de données non étiquetés décrit par deux attributs numériques. Puis, dans l'image du bas on observe ce même jeu de données auquel on ajoute une grille d'exemples artificiels à raison d'un exemple toutes les trois unités. On se retrouve donc avec un jeu de données enrichi par 49 nouveaux exemples. De plus, il est important de remarquer qu'à présent nous avons deux classes : les données de départ de la classe 1, et les données ajoutées de la classe 2.

Nous sommes alors en mesure de nous servir de cet enrichissement du jeu de données pour appliquer notre méthode de détection de sous-classes. La figure 4, image du haut, montre les graphes de liens que nous obtenons. L'image du bas, quant à elle, montre l'affectation en sous-classes qui est réalisé sur le jeu de données initial. On constate qu'on obtient 6 sous-classes dont 3 qui correspondent quasi exactement aux clusters que notre intuition nous pousserait à lister.

Cette méthode comportent des avantages et des inconvénients. L'avantage majeur est que, contrairement à beaucoup de méthodes de clustering existantes, il n'y a pas besoin de passer en paramètre un nombre de clusters prédéfini. Cependant, il faut choisir une « bonne » taille de grille. On a également la possibilité de ne pas s'inquiéter de la taille de la grille et de plutôt jouer sur le paramètre de tolérance de notre méthode. Dernier inconvénient : en cas de jeu de données en grande dimension, le nombre d'exemples « fantômes » à ajouter devient élevé, ce qui augmente le temps de calcul.

Tous ces points méritent plus d'investigations. Rappelons que ces travaux figurent en perspective de notre manuscrit car nous n'avons pas lancé suffisamment de tests sur des données réelles du domaine du clustering. Nous comptons nous y employer dans le futur.

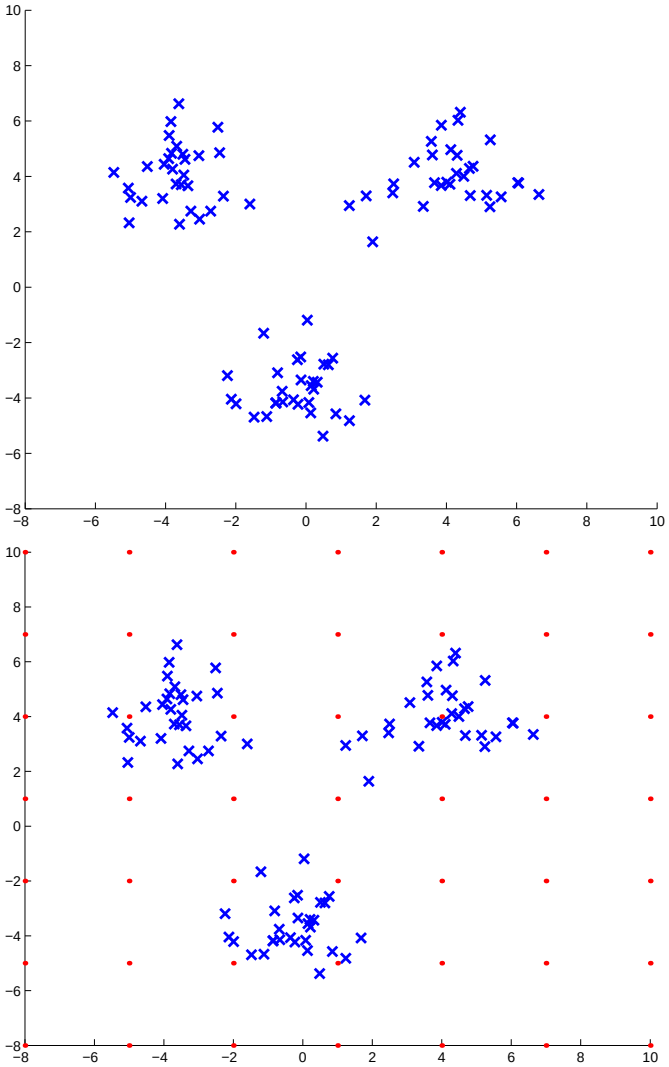

FIGURE 3 – Les données à « clusteriser » sont représentées en haut. A ce jeu de données on ajoute une grille d'exemples artificiels, image du bas.

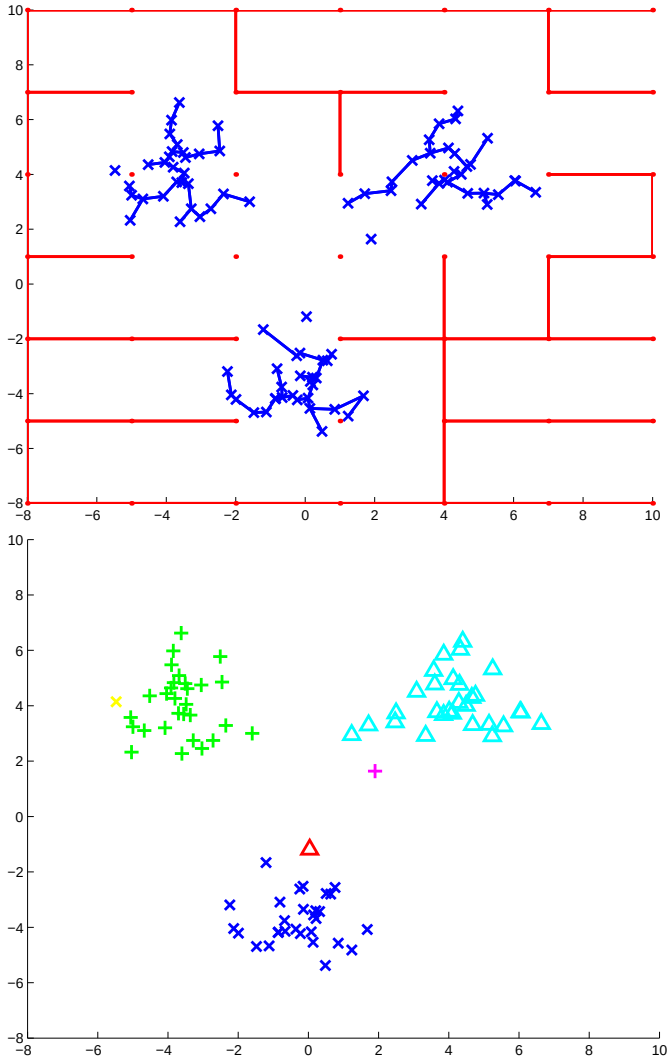

FIGURE 4 – Application de notre méthode au nouveau jeu de données. L'image du haut montre les graphes de liens d'amitié que nous obtenons. En bas se trouve l'affectation en sous-classes réalisée par notre approche.

5 Passage au flou

Enfin, la dernière perspective qui nous avons à l'esprit est le passage de notre approche d'un cadre « crisp » à un cadre « flou ». En effet, notre algorithme pourrait être adapté à des jeux de données où les exemples appartiennent à plusieurs catégories avec des degrés plus ou moins forts. Grâce au paramètre de tolérance et aux graphes d'amis pondérés par les distances, nous pourrions proposer des liens à certaines sous-classes plus prononcés qu'à d'autres.

Pour le moment nous considérons qu'un exemple appartient à une sous-classe s'il existe un lien entre lui et un autre exemple de la sous-classe. Mais nous pourrions rendre cette appartenance graduelle en fonction de la distance qui le sépare de l'exemple central ou de l'exemple le plus typique, par exemple.

Bibliographie

Aczel, J., & Oser, H. (1966). *Lectures on functional equations and applications.* Academic Press : New York.

Anand, R., Mehrotra, K., Mohan, C., & Ranka, S. (1995). Efficient classification for multiclass problems using modular neural networks. *Neural Networks, IEEE Transactions on*, *6*, 117–124.

Basu, S., Bilenko, M., & Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. *KDD '04 : Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 59–68). New York, NY, USA : ACM.

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algoritms.* New York : Plenum Press.

Bhattacharyya, S. (1999). Direct marketing performance modeling using genetic algorithms. *INFORMS Journal of computing*, *11*, 248–257.

Bosc, P., Dubois, D., Pivert, O., & Prade, H. (2000). Résumés de données et ensembles flous - principe d'une nouvelle approche. *Actes Rencontres Francophones sur la Logique Floue et ses Applications (LFA'2000), La Rochelle, 18/10/00-20/10/00* (pp. 333–340). Toulouse : Cepadues-Editions.

Bouchon-Meunier, B. (2007). *La logique floue.* Que sais-je ? Presses universitaires de France. Quatrième édition.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees.* Monterey, CA : Wadsworth and Brooks.

Bult, J. (1993). *Target selection for direct marketing.* Phd thesis, Rijkuniversiteit Groningen.

Calvo, T., & Mesiar, R. (1999). Generalized medians. *Proceedings of AGGREGATION 99* (pp. 159–165). Palma de Mayorca.

CNRTL (2009). Centre national de ressources textuelles et lexicales. (http ://www.cnrtl.fr).

Cormen, T., Leiserson, C., & Rivest, R. (1994). *Introduction à l'algorithmique*. Dunod.

Dale, E., & James, S. (2003). Set covering machine - learning with data balls. Transparents de présentation.

Detyniecki, M. (2000). *Mathematical aggregation operators and their application to video querying*. Thèse de doctorat, Université de Paris VI.

Detyniecki, M. (2001). Numerical aggregation operators : State of the art. *International Summer School on Aggregation Operators and their Applications*.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification*. Wiley-Interscience Publication.

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

Eiben, A. E., Euverman, T. J., Kowalczyk, W., Peelen, E., Slisser, F., & Wesseling, J. A. M. (1996). Comparing adaptive and traditional techniques for direct marketing. *Proceedings of the 4th European Congress on Intelligent Techniques and Soft Computing* (pp. 434–437). Verlag Mainz.

Eick, C. F., Vaezian, B., Jiang, D., & Wang, J. (2006). Discovering of interesting regions in spatial data sets using supervised cluster. *in PKDD 06, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*.

Eick, C. F., & Zeidat, N. (2005). Using supervised clustering to enhance classifiers. *in Proc. 15th International Symposium on Methodologies for Intelligent Systems (ISMIS* (pp. 248–256).

Fayyad, M. (1996). *Advances in knowledge discovery and data mining*. AAAI/MIT Press.

Forest, J., & Rifqi, M. (2009). Mise en évidence et caractérisation par prototype de sous-classes dans des classes non-homogènes. *Rencontres des Jeunes Chercheurs en Intelligence Artificielle*.

Forest, J., Rifqi, M., & Bouchon-Meunier, B. (2006a). Class segmentation to improve fuzzy prototype construction : Visualization and characterization of non homogeneous classes. *Fuzzy Systems, 2006 IEEE International Conference on*, 555–559.

Forest, J., Rifqi, M., & Bouchon-Meunier, B. (2006b). Segmentation de classes pour l'amélioration de la construction de prototypes flous : visualisation et caractérisation de classes non-homogènes. *Rencontres francophones sur la Logique Floue et ses Applications*, 29–36.

Gallinari, P. (1998). Modular neural net systems, training of. *The handbook of brain theory and neural networks*, *1*, 582–585.

Garey, M., & Johnson, D. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness.* W. H. Freeman.

George, R., & Srikanth, R. (1996). *Data summarization using genetic algorithms and fuzzy logic.* Genetic Algorithms and Soft Computing. Physica-Verlag, Heidelberg.

Guan, S.-U., & Zhu, F. (2004). Class decomposition for ga-based classifier agents - a pitt approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part B : Cybernetics*, *34*, 381–392.

Haughton, D., & Oulabi, S. (1993). Direct marketing modeling with cart and chaid. *Journal of Direct Marketing*, *7*, 16–26.

Hoffmann, A. G., Kwok, R. B. H., & Compton, P. (2001). Using subclasses to improve classification learning. *ECML '01 : Proceedings of the 12th European Conference on Machine Learning* (pp. 203–213). London, UK : Springer-Verlag.

Hughes, A. M. (2003). Rfm : ? Is it kudzu or is it gold ? *Database Marketing institute.*

Japkowicz, N. (2002). Supervised learning with unsupervised output separation. *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing* (pp. 321–325).

Kacprzyk, J. (1999). Fuzzy logic for linguistic summarization of databases. *Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE '99. 1999 IEEE International* (pp. 813–818 vol.2).

Kacprzyk, J., & Zadrozny, S. (2003). Linguistic summarization of data sets using association rules. *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on* (pp. 702–707 vol.1).

Kaymak, U. (2001). Fuzzy target selection using rfm variables. *Proceedings of Joint 9th IFSA World Congress and 20th NAFIPS International Conference* (pp. 1038–1043).

Kaymak, U., & Setnes, M. (2000). Target selection based on fuzzy clustering : a volume prototype approach to coil challenge 2000. *P. van der Putten and M. van Someren, editors, CoIL Challenge 2000 : The Insurance Company Case. Leiden Institute of Advanced Computer Science.*

Kolmogorov, A. (1930). Sur la notion de moyenne. *Atti delle Reale Accademia Nazionale dei Lincei Mem. Cl. Sci. Mat. Natur. Sez.* (pp. 323–343).

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* (pp. 48–50).

Lesot, M.-J. (2005a). *Classification non supervisée pour la visualisation de données structurées et la construction de prototypes.* Phd thesis, Université Paris 6.

Lesot, M.-J. (2005b). Similarity, typicality and fuzzy prototypes for numerical data. *6th European Congress on Systems Science, Workshop "Similarity and resemblance".*

Li, J., & Wang, J. (2003). Automatic linguistic indexing of pictures by a statistical modeling approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 25*, 1075–1088.

Ling, C., & Li, C. (1998). Data mining for marketing : Problems and solutions. *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining* (pp. 73–79).

Lu, B.-L., & Ito, M. (1999). Task decomposition and module combination based on class relations : A modular neural network for pattern classification. *IEEE-Neural Network, 10*, 1244.

Macqueen, J. B. (1967). Some methods of classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281–297).

Madeira, S., & Sousa, J. M. (2002). Comparison of target selection methods in direct marketing. *Proc. of European Symposium on Intelligent Technologies, Hybrid Systems and their Implementation on Smart Adaptive Systems, Eunite'02* (pp. 333–338).

Marchand, M., & Shawe-Taylor, J. (2001). Learning with the set covering machine. *Proc. 18th International Conf. on Machine Learning* (pp. 345–352). Morgan Kaufmann, San Francisco, CA.

Marchand, M., & Taylor, J. S. (2003). The set covering machine. *The Journal of Machine Learning Research, 3*, 723–746.

Marsala, C., & Detyniecki, M. (2006). University Paris 6 at TRECVID 2006 : Forests of fuzzy decision trees for high-level feature extraction. *TREC Video Retrieval Evaluation Online Proceedings.*

Marsh, D. (2001). Using the logistic procedure to model responses to financial services direct marketing. *NESUG Statistics, Data Analysis and Econometrics, 15.*

MediaDICO (2009). Site internet de dictionnaires. (http ://dictionnaire.mediadico.com).

Pijls, W., Potharst, R., & Kaymak, U. (2001). Pattern-based target selection applied to fund raising. *W. Gersten and K. Vanhoof, editors, Data Mining for Marketing Applications, ECML/PKDD-2001 Workshop*, 15–24.

Poel, D. V. D. (2003). Predicting mail-order repeat buying : Which variables matter ? *Working Papers of Faculty of Economics and Business Administration, Ghent University.*

Potharst, R., Kaymak, U., & Pijls, W. (2002). Neural networks for target selection in direct marketing. *Kate Smith and Jatinder Gupta, editors, Neural Networks in Business : techniques and applications*, 89–110.

Putten, P. V. D. (1999). Data mining in direct marketing databases. *Walter Baets (ed). Complexity and Management : A Collection of Essays. World Scientific Publishers.*

Quinlan, R. J. (1993). *C4.5 : Programs for machine learning (Morgan Kaufmann series in machine learning).* Morgan Kaufmann.

Raschia, G. (2001). *Utilisation de la logique floue pour la génération de résumés : Application aux bases de données relationnelles.* Thèse de doctorat, Université de Nantes.

Rasmussen, D., & Yager, R. (1997). Summarysql-a fuzzy tool for data mining. *Intelligent Data Analysis-An International Journal*, *1*.

Rifqi, M. (1996a). Constructing prototypes from large databases. *Information Processing and Management of Uncertainty (IPMU'96).*

Rifqi, M. (1996b). *Mesures de comparaison, typicalité et classification d'objets flous : théorie et pratique.* Phd thesis, Université Paris 6.

Rifqi, M., Berger, V., & Bouchon-Meunier, B. (2000). Discrimination power of measures of comparison. *Fuzzy Sets and Systems*, *110*, 189–196.

Rosch, E. (1978). Principles of categorization. *E. Rosch and B. Lloyd, editors, Cognition and categorization*, 27–48.

Sellers, J., & Hughes, A. M. (2003). Rfm migration analysis. a new approach to a proven techniques. *Database Marketing institute.*

Silvert, W. (1979). Symmetric summation : a class of operations on fuzzy sets. *IEEE Transactions on Systems, Man and Cybernetics*, *9*, 659–667.

Sousa, J., Kaymak, U., & Madeira, S. (2002). A comparative study of fuzzy target selection methods in direct marketing. *Proceedings of 2002 IEEE International Conference on Fuzzy Systems* (pp. 1251–1256).

UCI (2009). Machine learning repository. http ://www.ics.uci.edu/ mlearn/MLRepository.html.

Université de Waikato, N. Z. (2009). Weka. http ://www.cs.waikato.ac.nz/ml/weka/.

Vilalta, R., Achari, M.-K., & Eick, C. (2003). Class decomposition via clustering : a new framework for low-variance classifiers. *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on* (pp. 673–676).

Vilalta, R., & Rish, I. (2003). A decomposition of classes via clustering to explain and improve Naive Bayes. *Proceedings of European Conference on Machine Learning* (pp. 444–455).

Wang, J., Neskovic, P., & Cooper, L. N. (2005). Pattern classification via single spheres. *Lecture Notes in Computer Science*, *3735*, 241–252.

Wang, J., Neskovic, P., & Cooper, L. N. (2006). A minimum sphere covering approach to pattern classification. *International Conference on Pattern Recognition*, *3*, 433–436.

Wheaton, J. (1996). Rfm cells : The kudzu of segmentation. *DM News*.

Wu, J., Xiong, H., Wu, P., & Chen, J. (2007). Local decomposition for rare class analysis. *KDD '07 : Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 814–823). New York, NY, USA : ACM.

Yager, R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *Systems, Man and Cybernetics, IEEE Transactions on*, *18*, 183–190.

Yager, R., & Rubinson, T. C. (1981). Linguistic summaries of data bases. *20th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, *20*, 1094–1097.

Yager, R. R. (1982). A new approach to the summarization of data. *Information Sciences*, *28*, 69–86.

Yager, R. R. (1991). On linguistic summaries of data. In W. Frawley and G. Piatetsky-Shapiro (Eds.), *Knowledge discovery in databases*, 347–366. MIT Press.

Zadeh, L. (1965). Fuzzy sets. *Information Control*, *8*, 338–353.

Zadeh, L. (1975). The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, *1*, 119–249.

Zadeh, L. (1983). A computational approach to fuzzy quantifiers in natural languages. *International series in modern applied mathematics and computer science*, *5*, 149–184.

Zahavi, J., & Levin, N. (1997). Applying neural computing to target marketing. *Journal of Direct Marketing*, *11*, 5–22.

Annexes

1 Opérateurs d'agrégation

Cette partie sur les opérateurs d'agrégation est issue d'un rapport réalisé dans le cadre du projet Infom@gic. "*Le projet Infom@gic est coordonné par la société Thalès et regroupe plus de 20 partenaires, aussi bien privés (EADS, Pertimm, Xerox, Arisem, TEMIS, ...) que publics (CEA, INA, Laboratoires de recherche d'universités françaises, ...). Il est soutenu par le pôle de compétitivité francilien Cap Digital "Image, Multimédia et Vie Numérique (IMVN)", un groupement de 30 grandes entreprises et plus de 200 PME centré sur les nouveaux marchés des contenus numériques.*"

"L'objectif est de sélectionner, de tester, d'intégrer et de valider des applications opérationnelles dans le domaine de la recherche, de l'indexation et de l'extraction de connaissances ainsi que de la fusion d'informations multimédia (image, son, texte et données structurées)."

Les opérateurs d'agrégation sont de très bon outils de caractérisation car ils fournissent un moyen de *condenser l'information* et d'en *synthétiser le contenu*.

Dans cette partie nous nous concentrerons sur les différents opérateurs d'agrégation. Nous commencerons par donner une définition formelle de l'agrégation, puis nous poursuivrons en énumérant les propriétés mathématiques que l'on peut exiger des opérateurs. Enfin nous dresserons un panorama de ces opérateurs en nous inspirant des travaux de Detyniecki (2000) et Detyniecki (2001).

1.1 Définition mathématique de l'agrégation

Intuitivement l'agrégation se conçoit comme le fait de regrouper n objets d'un ensemble donné en un unique objet qui appartient également à cet ensemble. Dans un cadre plus formel de l'agrégation, les objets qu'on agrège sont des nombres réels et les opérateurs sont des fonctions qui, à tout n-uplet $(x_1, x_2, ..., x_n)$ de nombre réels, associent un nombre réel y :

$$y = Agreg(x_1, x_2, ..., x_n)$$

Evidemment la fonction *Agreg* doit satisfaire un certain nombre de conditions pour justifier le nom d'opérateur d'agrégation. Bien qu'il existe plusieurs définitions dans la littérature celle qui est communément admise est la suivante : Une fonction *Agreg* est une fonction l'agrégation si et seulement si :

i $Agreg : [0,1]^n \rightarrow [0,1]$

ii $Agreg(x) = x$

iii $Agreg(x_1, x_2, ..., x_n) \leq Agreg(y_1, y_2, ..., y_n)$ si $(x_1, x_2, ..., x_n) \leq (y_1, y_2, ..., y_n)$

iv $Agreg(0, ..., 0) = 0$ et $Agreg(1, ..., 1) = 1$

Espaces de définition

$$Agreg : [0,1]^n \rightarrow [0,1]$$

La première condition nous dit qu'on travaille dans des intervalles réels bornés. Même si, la définition classique se focalise sur l'intervalle unitaire, on peut utiliser un intervalle quelconque. Aussi on comprend qu'une fonction d'agrégation prend n réels en entrée et a pour image un réel, qui est le résultat de l'agrégation. Ce résultat est borné en raison des conditions (iii) et (iv).

Agrégation d'un seul réel :

$$Agreg(x) = x$$

La deuxième condition nous dit que l'agrégation d'un élément tout seul est lui-même. Cette propriété a son importance en tant que point de départ axiomatique.

Monotonie :

$$Agreg(x_1, x_2, ..., x_n) \leq Agreg(y_1, y_2, ..., y_n) \; si \; (x_1, x_2, ..., x_n) \leq (y_1, y_2, ..., y_n)$$

La propriété de monotonie est importante chez un opérateur d'agrégation : on s'attend à ce que si un argument augmente alors l'agrégation augmente à son tour.

Cas limites : Le type de comportement d'un opérateur dans le cas limites est lourd de conséquences. Pour qu'un opérateur soit dit d'agrégation, il doit satisfaire les propriétés suivantes :

$$Agreg(0, ..., 0) = 0 \; et \; Agreg(1, ..., 1) = 1$$

Cette propriété, combinée avec la précédente, font que le résultat d'une agrégation est forcement dans l'intervalle $[0, 1]$.

Ce sont les quatre conditions fondamentales, qui définissent bien ce qu'on attend d'un opérateur d'agrégation : qu'à partir de n réels entre 0 et 1, il nous donne forcement un réel entre 0 et 1 qui sera le résultat de l'agrégation. Un opérateur d'agrégation typique est la moyenne.

Mis à part les propriétés fondamentales, d'autres propriétés peuvent être souhaitées et donner à l'opérateur le comportement qui conviendra à une application spécifique. Ces nouvelles propriétés vont garantir un comportement spécifique, qui impliquera une interprétation correspondante : opérateur de type moyenne, opérateur logique, etc. Dans la section suivante nous allons donner une liste de ces propriétés.

1.2 Propriétés des opérateurs

Continuité

On peut vouloir exiger d'un opérateur d'agrégation qu'il respecte la continuité. Cela permet de garantir une certaine robustesse. En effet la continuité nous garantit qu'il n'y a pas de sauts quand les valeurs à agréger varient faiblement.

Associativité

L'associativité est la propriété qui garantit que les informations à agréger peuvent l'être par paquets sans changer le résultat final.

$$Agreg(x_1, x_2, x_3) = Agreg(Agreg(x_1, x_2), x_3) = Agreg(x_1, Agreg(x_2, x_3))$$

C'est particulièrement intéressant pour les cas où l'opérateur ne peut prendre en compte que deux éléments.

Commutativité

Cette propriété garantit que les valeurs peuvent être agrégées dans n'importe quel ordre sans changer le résultat final.

$$Agreg(x_1, x_2) = Agreg(x_2, x_1)$$

Idempotence

Cette propriété implique qu'en agrégeant n fois la même valeur on retrouve cette valeur. Cette propriété est classiquement utilisée si on souhaite que le résultat de l'agrégation soit

un élément représentatif de l'ensemble agrégé.

$$Agreg(x, x, ..., x) = x$$

Principe de Pareto

Cette propriété borne le résultat de l'agrégation entre le minimum et le maximum des valeurs agrégées. Dans le même esprit que précédemment si on cherche un opérateur de compromis, on demandera à respecter le principe de Pareto.

$$\min_{i=1}^{n}(x_i) \leq Agreg(x_1, ..., x_n) \leq \max_{i=1}^{n}(x_i)$$

Elément neutre

Si l'opérateur admet un élément neutre cela signifie que le résultat de l'agrégation ne change pas que l'élément soit présent ou non. Dans le cas d'un opérateur de type compromis, il peut être considéré comme une abstention. Dans le cas des opérateurs logiques il définit le type d'opérateur.

$$Agreg^{[n]}(x_1, ..., x_{i-1}, e, x_{i+1}, ..., x_n) = Agreg^{[n-1]}(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$$

Elément absorbant

Si l'opérateur admet un élément absorbant, cela signifie que le résultat de l'agrégation est cet élément dès qu'il est présent. Dans le cas d'un opérateur de type compromis, il peut être considéré comme un score éliminatoire ou un veto.

$$Agreg(x_1, ..., a, ..., x_n) = a$$

1.3 Principaux opérateurs d'agrégation

Les opérateurs d'agrégation peuvent être définis, soit de manière constructive à partir des propriétés précédemment annoncées ou bien de manière descriptive, par une définition mathématique et ensuite décrits par les propriétés.

Il faut noter que tous les nouveaux opérateurs sont typiquement le résultat de la relaxation de certaines propriétés d'un opérateur connu.

Dans ce qui suit nous présentons une liste non exhaustive des opérateurs les plus souvent rencontrés.

Opérateurs de type compromis

Les opérateurs d'agrégation les plus communément utilisés sont ceux qui proposent comme résultat une valeur "représentative" de l'ensemble de réels agrégés. Car ces valeurs sont dispersées dans l'intervalle $[0, 1]$, ces opérateurs doivent avoir un comportement de type compromis, compensant les valeurs élevées avec les plus faibles. La valeur résultante est typiquement centrale. Pour garantir un comportement de ce type il est typiquement demandé que la propriété d'idempotence soit vérifiée.

Moyenne arithmétique

On peut dire qu'il s'agit de l'opérateur d'agrégation le plus connu. Il est souvent utilisé car, d'une part il est très simple, et d'autre part il vérifie un grand nombre de propriétés garantissant axiomatiquement que le résultat de l'agrégation est représentatif par compromis. Ces propriétés sont entre autres : monotonie, continuité, symétrie, associativité, idempotence et surtout la stabilité aux changements d'échelle (linéaire). Il n'admet ni élément neutre, ni élément absorbant.
Moyenne arithmétique :

$$M(x_1, ..., x_n) = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Il existe une version pondérée, pour laquelle un certain nombre de propriétés sont perdues, mais qui a l'avantage de permettre une déformation linéaire de chacune des dimensions de l'espace, ce qui se traduit dans la pratique par une pondération de l'importance de chaque critère agrégé.
Moyenne pondérée :

$$M_{w_1,...,w_n}(x_1, ..., x_n) = \sum_{i=1}^{n} w_i \cdot x_i$$

Moyennes quasi-arithmétiques

Toute une série d'extensions qui déforment l'espace uniformément, appelées moyennes quasi-arithmétiques, ont été étudiées par Kolmogorov (1930) et Aczel et Oser (1966) plus récemment.

$$M_f(x_1, ..., x_n) = f^{-1}\left[\frac{1}{n} \sum_{i=1}^{n} f(x_i)\right] = f^{-1}\left[\sum_{i=1}^{n} \left(\frac{1}{n} \cdot f(x_i)\right)\right]$$

Le choix de la fonction f donne la transformation de l'espace, qui implique des sensitivités à des variations différentes dépendant des valeurs agrégées et de son résultat. En choisissant différentes fonctions f on obtient des opérateurs bien connus :

Moyenne géométrique :

$$M_{geometrique}(x_1, ..., x_n) = \left(\prod_{i=1}^{n} x_i \right)^{\frac{1}{n}}$$

Moyenne harmonique :

$$M_{harmonique}(x_1, ..., x_n) = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

Médiane

Un autre opérateur d'agrégation cherche à donner une valeur représentative : la médiane. Cet opérateur ordonne les valeurs par ordre croissant et renvoit la valeur du milieu. Il est intéressant puisqu'il vérifie les propriétés de monotonie, symétrie, idempotence, et surtout il permet d'atténuer l'influence perturbatrice des valeurs extrêmes enregistrées lors de circonstances exceptionnelles (ce qui n'est pas le cas pour la moyenne). Tout comme pour la moyenne, des opérateurs de type médiane plus généraux ont récemment été développés par Calvo et Mesiar (1999).

Somme symétrique

Les sommes symétriques sont une famille d'opérateurs d'agrégation qui sont continus et auto-duaux, c'est-à-dire qu'ils satisfont la contrainte suivante :

$$S(x_1, ..., x_n) = 1 - S(1 - x_1, ..., 1 - x_n)$$

Elles ont été étudiées en détail par Silvert (1979). On peut noter que, pour deux éléments, les sommes symétriques s'écrivent en fonction de G sous la forme :

$$S(x, y) = \frac{G(x, y)}{G(x, y) + G(1 - x, 1 - y)}$$

G est une fonction continue, croissante, positive, satisfaisant $G(0, 0) = 0$. De plus, il faut faire attention à utiliser la convention $0/0 = 1/2$. Généralement les sommes symétriques ne sont pas commutatives. Un bon exemple de cette famille d'opérateurs est la moyenne pondérée.

Opérateurs OWA (Ordered Weighted Averaging Operators)

Le concept d'Ordered Weighted Averaging Operator (OWA) fut initialement proposé par Yager (1988). C'est un opérateur de type compensation qui a une palette de comportements qui va du minimum au maximum, en passant par la moyenne arithmétique.

$$OWA(x_1, ..., x_n) = \sum_{i=1}^{n} w_i \cdot x_{\sigma(i)}$$

Où σ est une permutation qui ordonne les éléments à agréger de telle sorte que : $x_{\sigma(1)} \leq ... \leq x_{\sigma(n)}$. Les poids doivent tous être positifs ou nuls ($w_i \geq 0$) et leur somme est égale à l'unité ($\sum_{i=1}^{n} w_i = 1$). Le choix des poids w_i, permet une telle souplesse que la plupart des opérateurs précédents sont ainsi généralisés :

Opérateur	équivalent OWA
Minimum	$\begin{cases} w_1 = 1 \\ w_i = 0, \forall i \neq 1 \end{cases}$
Maximum	$\begin{cases} w_n = 1 \\ w_i = 0, \forall i \neq n \end{cases}$
Médiane	$\begin{cases} w_{\frac{n+1}{2}} = 1 \text{ si } n \text{ est impair} \\ w_{\frac{n}{2}} = \frac{1}{2} \text{ et } w_{\frac{n}{2}+1} = \frac{1}{2} \text{ si } n \text{ est pair} \\ w_i = 0 \text{ sinon} \end{cases}$
Moyenne arithmétique	$w_i = \frac{1}{n} \forall i$

2 Logique floue

La théorie des sous-ensembles flous a été développée par Zadeh (1965). Elle généralise la théorie des sous-ensembles « classiques » dans laquelle un objet appartient ou n'appartient pas à un sous-ensemble. En théorie des sous-ensembles flous, la notion d'appartenance est rendue beaucoup plus flexible et un objet appartient à un sous-ensemble avec un certain degré compris entre 0 et 1.

L'idée derrière ce formalisme est de permettre la modélisation des notions linguistiques telles que « une bonne note »ou « une mauvaise note ». Ces notions dans la théorie classique demanderaient de fixer des frontières faisant émerger des situations aberrantes. Par exemple, si on considère qu'une bonne note est supérieure ou égale à 15, un élève avec une note de $14,8/20$ n'entrerait pas dans la catégorie alors qu'un autre avec une note de

15 s'y trouverait.

2.1 Définition

Un sous-ensemble flou A est représenté par une fonction d'appartenance, qui associe à toute valeur du domaine D un degré d'appartenance :

$$\begin{aligned} \mu_A : D &\rightarrow [0,1] \\ x &\mapsto \mu_A(x) \end{aligned}$$

Alors que les sous-ensembles classiques admettent une fonction d'appartenance binaire $0, 1$, ici le degré d'appartenance est compris entre 0 et 1 où 0 est la non appartenance et 1 l'appartenance totale. Le plus souvent les fonctions d'appartenance sont de type trapézoïdal.

L'intérêt des fonctions d'appartenance est qu'elles permettent de modéliser les données imprécises et/ou incertaines. L'imprécision d'une valeur sera représentée comme une fonction formant un "pic" centré autour de la valeur. Une incertitude correspondra à une fonction dont le degré d'appartenance n'atteint pas des valeurs élevées.

2.2 Caractéristiques des sous-ensembles flous

La fonction d'appartenance μ d'un sous-ensemble flou A est caractérisée par trois éléments :

- son noyau : l'ensemble des points ayant un degré d'appartenance totale.

$$Noy_A = x \in D, \text{ tels que } \mu_A(x) = 1$$

- son support : l'ensemble des points ayant un degré d'appartenance non nul.

$$Supp_A = x \in D, \text{ tels que } \mu_A(x) \neq 0$$

- sa hauteur : la valeur maximale de la fonction d'appartenance.

$$h(A) = \max_{x \in D} \mu_A(x)$$

La plupart du temps on travaille avec des sous-ensembles flous normalisés, c'est à dire dont la hauteur est égale à 1.

2.3 T-normes et T-conormes

En logique classique on dispose d'opérateurs pour combiner les prédicats. Sans ceux-ci il n'est pas possible de construire des prédicats complexes et évolués. Il s'agit des opérateurs de conjonction et de disjonction.

La logique floue dispose elle aussi de ce type d'outils grâce aux $\top$-normes et aux $\perp$-conormes.

Une *t-norme* est une fonction $\top : [0,1] \times [0,1] \rightarrow [0,1]$ qui vérifie les propriétés suivantes :

- Commutativité : $\top(x,y) = \top(y,x)$.
- Monotonie (non décroissance) : $\top(x,y) \leq \top(u,v)$ si $x \leq u$ et $y \leq v$.
- Associativité : $\top(x,\top(y,z)) = \top(\top(x,y),z)$.
- L'unité comme élément neutre : $\top(x,1) = x$

De la même façon, une *t-conorme* vérifie les mêmes propriétés sauf que zéro est l'élément neutre : Une *t-conorme* est une fonction $\perp : [0,1] \times [0,1] \rightarrow [0,1]$ qui vérifie les propriétés suivantes :

- Commutativité : $\perp(x,y) = \perp(y,x)$.
- Monotonie (non décroissance) : $\perp(x,y) \leq \perp(u,v)$ si $x \leq u$ et $y \leq v$.
- Associativité : $\perp(x,\perp(y,z)) = \perp(\perp(x,y),z)$.
- L'unité comme élément neutre : $\perp(x,0) = x$

Il existe beaucoup d'opérateurs dans la littérature. Voici une série des plus usuels donnée par Bouchon-Meunier (2007) :

	T-norme	T-conorme
Zadeh	$\min(x,y)$	$\max(x,y)$
Probabiliste	$x \cdot y$	$x + y - xy$
Lukasiewicz	$\max(x+y-1,0)$	$\min(x+y,1)$
Hamacher ($\gamma > 0$)	$\frac{xy}{\gamma + (1-\gamma)(x+y-xy)}$	$\frac{xy}{\gamma + (1-\gamma)(x+y-xy)}$

3 Echantillon de la base d'images Corel 1000

Voici l'intégralité de la base de données que nous avons utilisé pour montrer, partie 2.3 page 105, que notre approche de découverte de sous-catégories permettait de construire

des prototypes efficaces dans le but de resumer une base d'images. Plus d'informations sur cette base sont disponibles dans l'article de Li et Wang (2003). La base se constitue de 10 classes de 100 images chacune. Les thèmes sont variés : Afrique, plages, monuments, bus, dinosaures, élephants, fleurs, chevaux, montagnes et nourriture.

FIGURE 1 – Les images de la classe Afrique.

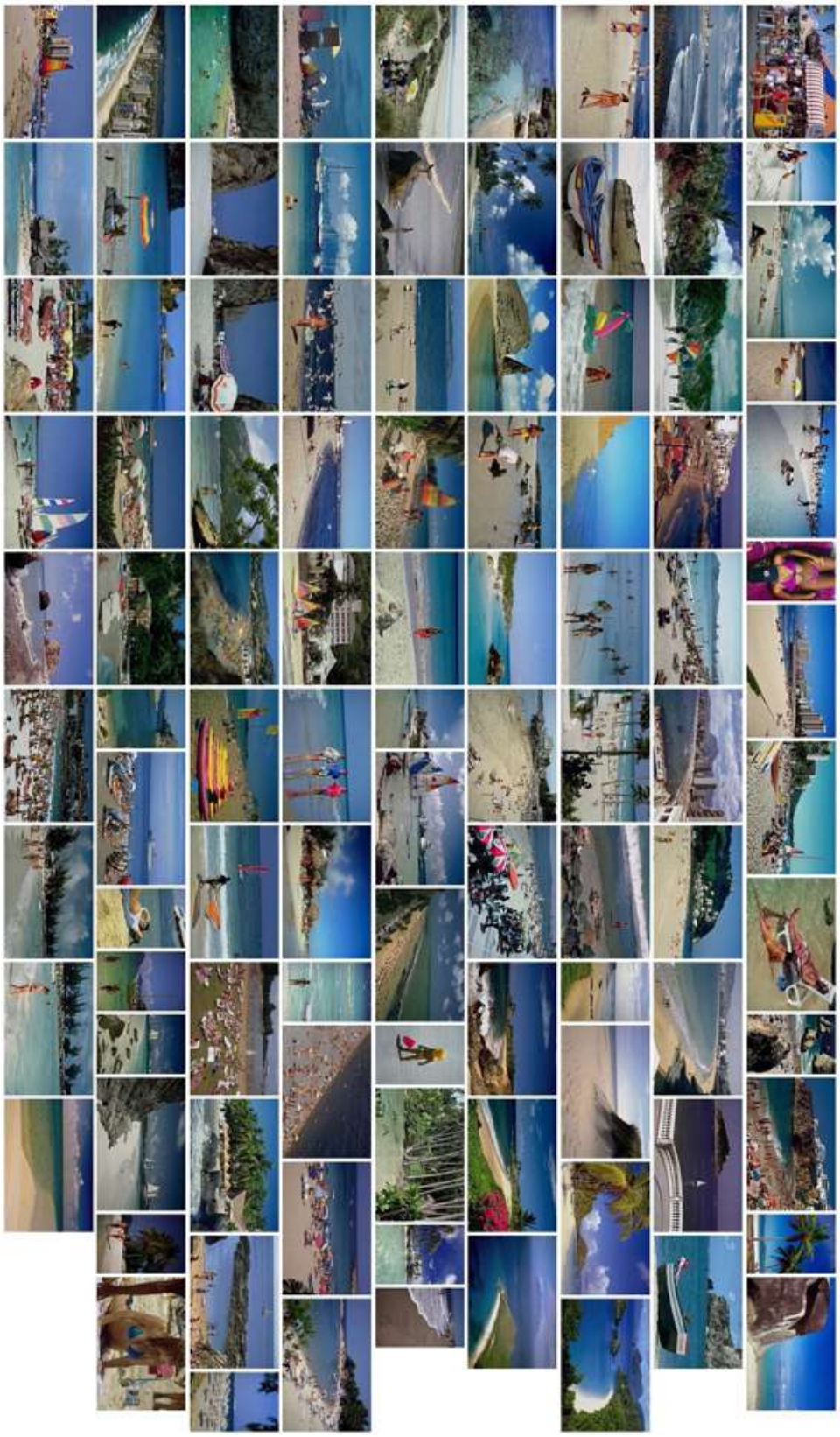

FIGURE 2 – Les images de la classe Plage.

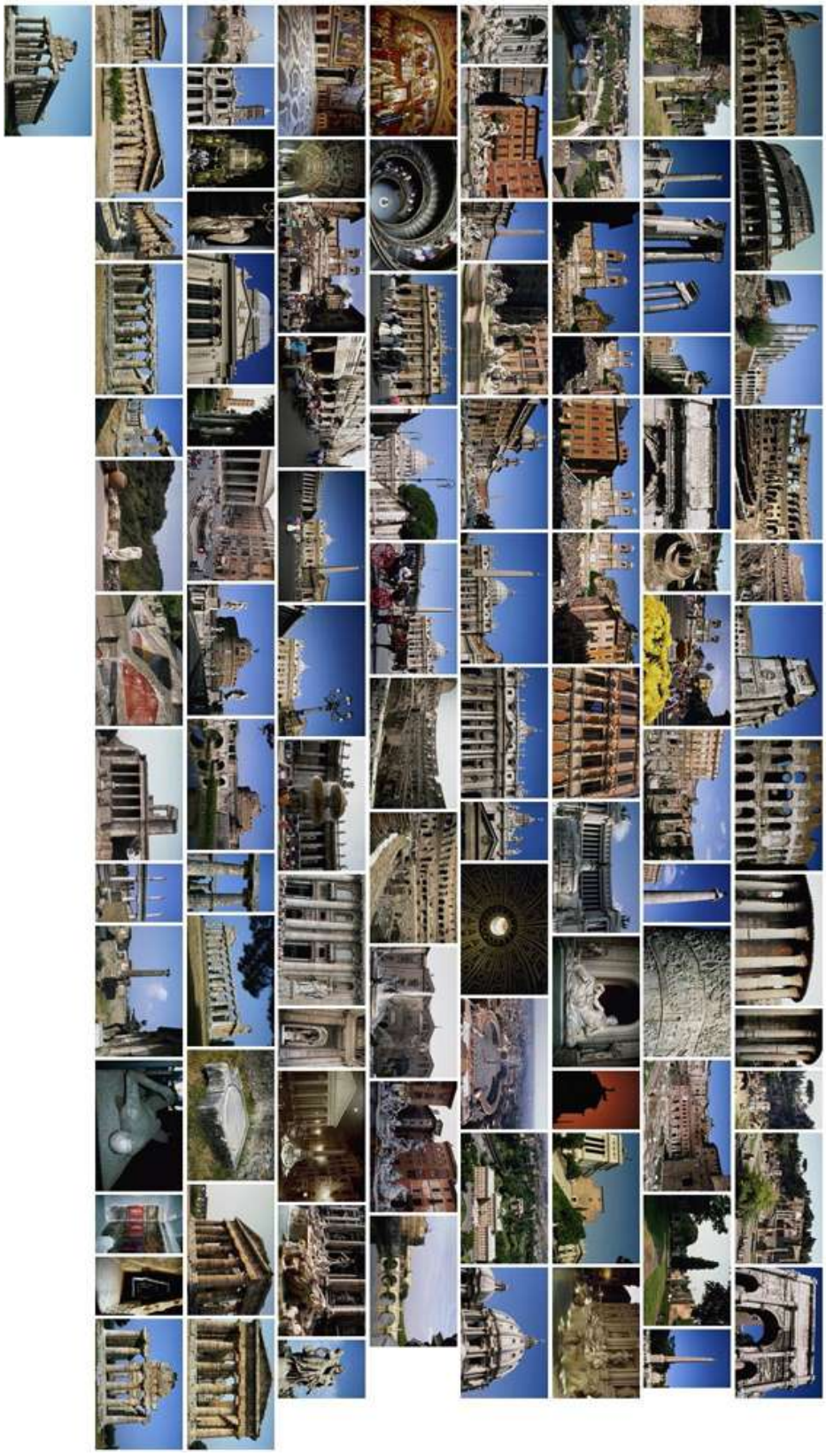

FIGURE 3 – Les images de la classe Monuments.

FIGURE 4 – Les images de la classe Bus.

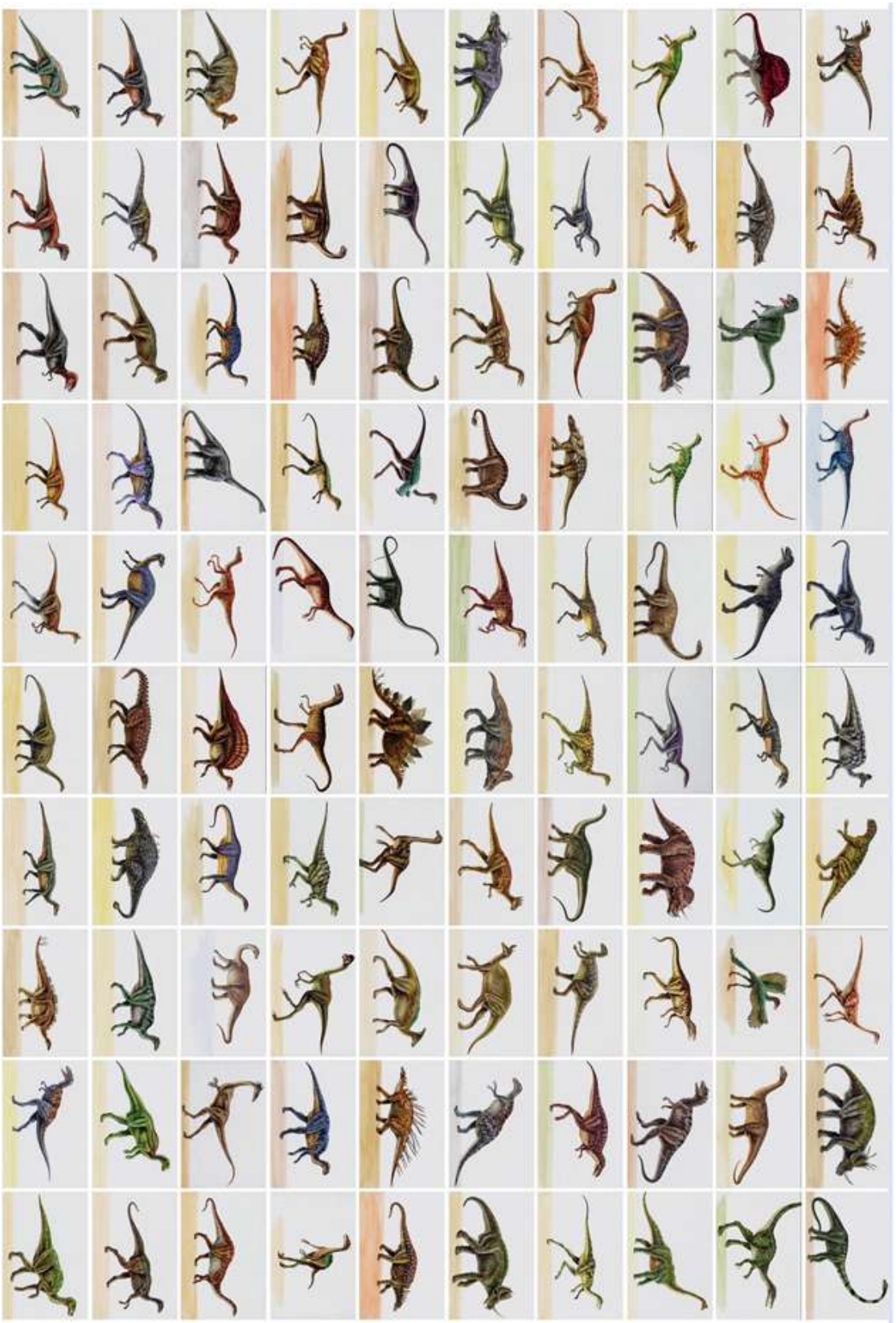

FIGURE 5 – Les images de la classe Dinosaures.

FIGURE 6 – Les images de la classe Elephants.

FIGURE 7 – Les images de la classe Fleurs.

FIGURE 8 – Les images de la classe Chevaux.

FIGURE 9 – Les images de la classe Montagne.

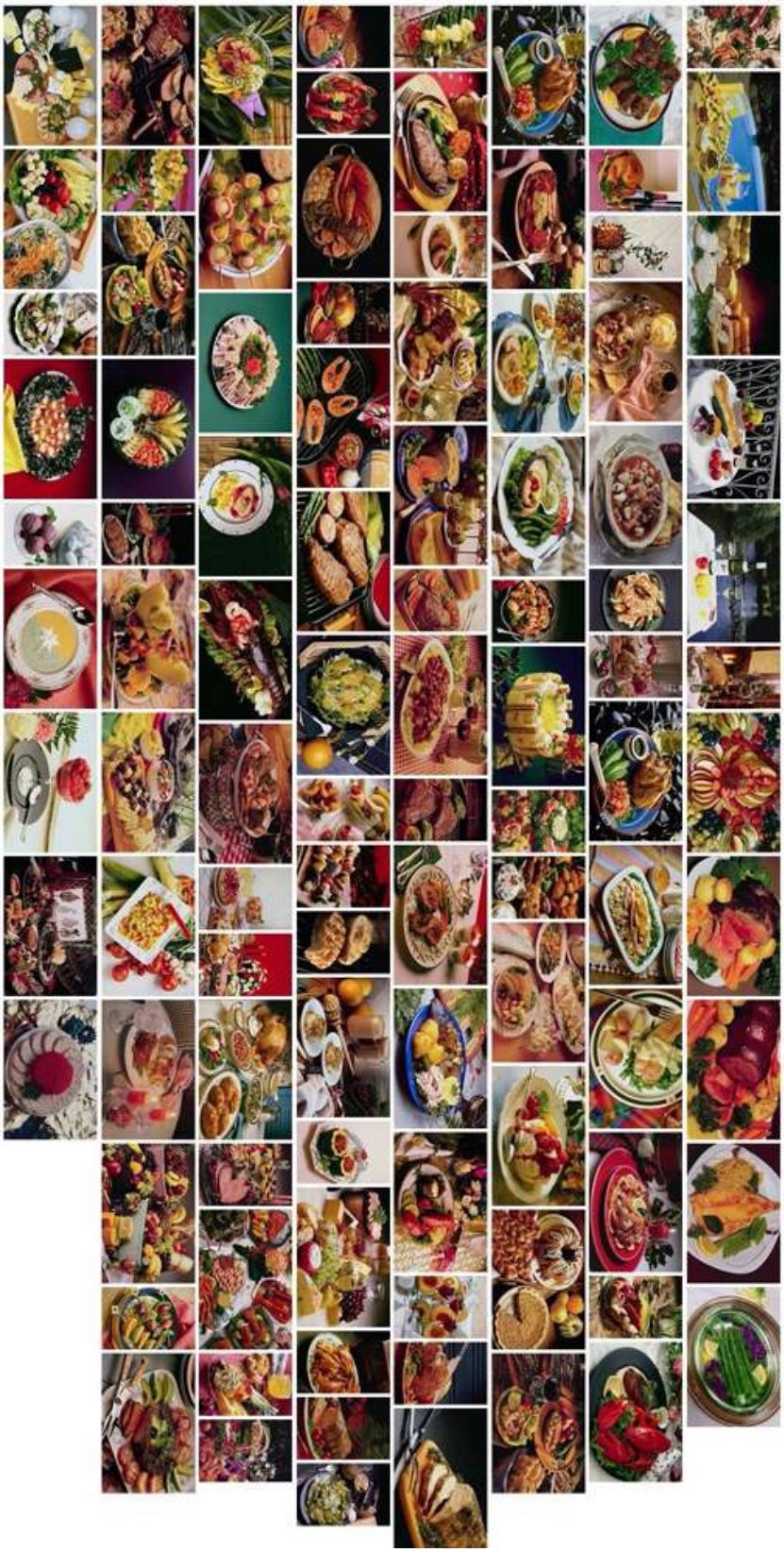

FIGURE 10 – Les images de la classe Nourriture.