Une méthode de classification non-supervisée pour l'apprentissage de règles et la recherche d'information

Guillaume Cleuziou

Une méthode de classification non-supervisée pour l'apprentissage de règles et la recherche d'information

THÈSE

présentée

pour l'obtention du grade de

Docteur de l'Université d'Orléans

Discipline Informatique

par

Guillaume Cleuziou

Soutenue le 8 décembre 2004

Membres du jury :

Richard Emilion	Professeur - Université d'Orléans	Examinateur
Lionel Martin	Maître de conférences - Université d'Orléans	Examinateur
Lorenza Saitta	Professeur - Université du Piemont - Italie	Rapporteur
Michèle Sebag	Directeur de recherche CNRS - Université Paris-Sud	Président du jury
Christel Vrain	Professeur - Université d'Orléans	Directrice de thèse
Djamel A. Zighed	Professeur - Université Lumière - Lyon 2	Rapporteur

Remerciements

Je tiens à remercier tout d'abord Lorenza Saitta et Djamel Zighed d'avoir accepté de rapporter cette thèse, pour le temps qu'ils auront bien voulu y consacrer. Qu'ils soient certains que je considère leur contribution comme un honneur.

Mes remerciements vont également à Michèle Sebag et Richard Emilion ; en participant au jury de thèse, ils apportent un remarquable soutien à mon travail.

Je remercie également Christel Vrain, directrice de cette thèse. Son expérience et sa rigueur scientifique m'auront été d'une grande aide tant dans la progression de mes travaux de recherche que dans l'écriture de ce mémoire.

Un grand merci à Lionel Martin qui m'a accompagné pendant ces quatre années, du stage de DEA jusqu'à la soutenance de thèse. Je le remercie pour sa patience, sa compétence et sa modestie. Ce fut un réel plaisir de travailler ensemble et je souhaite vivement poursuivre cette collaboration.

Je n'oublie pas toutes les personnes qui, scientifiquement ou socialement ont contribué à la réussite de ce travail : Viviane et Céline pour m'avoir fait partager leurs préoccupations linguistiques ; Sylvie, Souley, Agnès, Armelle, Chantal et Ansaf pour les discussions enrichissantes que nous avons échangées et les conseils avisés qui s'en dégageaient. Merci également aux membres du LIFO pour leur accueil et à ses responsables pour m'avoir permis cet épanouissement personnel en m'orientant sur le chemin de la recherche.

Sur un plan plus personnel c'est à Ludivine que vont mes premiers remerciements, elle a su s'intéresser à mon travail et "soigner" quelques moments de découragements. Pour son accompagnement et sa patience, notamment durant la période de rédaction de ce manuscrit, je la remercie.

Je remercie également mes parents pour leur soutien tout au long de ces années d'études, mes frères David et Christophe et mes amis Youssef, Baptiste, Guillaume, Antoine, Sam, Damien, Cathy et Linda pour l'intérêt qu'ils ont manifesté à l'égard de mon travail.

Table des matières

Introduction **1**

1 État de l'art du Clustering **5**
1.1 Clustering et apprentissage 7
1.2 Cadre formel du clustering 7
1.2.1 Partitions, pseudo-partitions et partitions floues 8
1.2.2 Hiérarchies et pseudo-hiérarchies 8
1.2.3 Centroïdes et médoïdes 9
1.2.4 Concavité vs. convexité d'un cluster 10
1.2.5 Les outliers 10
1.2.6 Autres notions utiles 11
1.3 Les trois principales étapes du clustering 11
1.3.1 La préparation des données 12
1.3.2 Le choix de l'algorithme 13
1.3.3 L'exploitation des clusters 15
1.4 La similarité 15
1.4.1 Propriétés formelles de base 15
1.4.2 Similarité et variables numériques 17
1.4.3 Similarité et variables symboliques 17
1.5 Clustering "dur", clustering avec recouvrements et clustering "flou" 19
1.6 Les différentes méthodes de clustering 20
1.6.1 Le clustering hiérarchique 20
1.6.2 Le clustering par partitionnement 27
1.6.3 Le clustering par mélange de densités de probabilités 32
1.6.4 Le clustering par grilles 34
1.6.5 Le clustering par densités 38
1.6.6 Le clustering conceptuel 41
1.6.7 Quelques autres approches de clustering 45
1.7 Techniques d'évaluation du clustering 46
1.7.1 Critères externes 48
1.7.2 Critères internes 48
1.7.3 Critères relatifs 49
1.7.4 Critères d'évaluation pour clustering flou 50

1.7.5 Adaptation au clustering avec recouvrements 50
1.8 Conclusion sur le clustering 51

2 L'algorithme de clustering PoBOC 53
2.1 Motivations et cadre général de l'algorithme 54
2.2 Présentation de l'algorithme PoBOC 54
2.2.1 Présentation générale de PoBOC 54
2.2.2 Présentation formelle de l'algorithme PoBOC 56
2.3 Discussion sur l'algorithme PoBOC 65
2.3.1 Rappel du processus global 65
2.3.2 Positionnement de l'algorithme PoBOC 66
2.3.3 Traitement de grandes bases 68
2.4 Premières expérimentations 69
2.4.1 Analyse de PoBOC sur la base de données Iris 69
2.4.2 Évaluation de PoBOC sur d'autres bases de données 75
2.5 Conclusion 78

3 Apprentissage de règles de classification par regroupement d'objets 79
3.1 Introduction à l'apprentissage supervisé 80
3.2 Apprentissage d'un ensemble de règles 80
3.2.1 Le système AQ [123] 83
3.2.2 Le système CN2 [24] 83
3.2.3 Le système pFOIL [150] 84
3.2.4 Le système RISE [48] 84
3.3 Motivation de l'approche par regroupement 85
3.3.1 Le problème du choix d'un "bon" sélecteur 85
3.3.2 Utilisation du clustering dans un contexte supervisé 86
3.3.3 L'approche générale RuLe-Clust 87
3.3.4 Vers un pseudo-partitionnement des classes 88
3.4 Le système RuLe-Clust 88
3.4.1 Stratégie générale du classifieur 90
3.4.2 Test d'existence d'une règle unique 90
3.4.3 Algorithme de construction d'une règle 92
3.4.4 Variantes relatives à l'algorithme de clustering utilisé 92
3.5 Évaluation du système RuLe-Clust 95
3.5.1 Présentation de l'étude 95
3.5.2 Évaluation du principe général de décomposition 96
3.5.3 Évaluation de l'algorithme PoBOC 97
3.5.4 Influence d'une pseudo-partition 99
3.5.5 Comparaison avec d'autres classifieurs 100
3.6 Conclusion 101

4 Apprentissage disjonctif par décomposition de concepts 103
4.1 Introduction 104
4.2 Bases de la logique du premier ordre 104
4.3 Apprentissage disjonctif 105
4.3.1 Le système FOIL 106
4.3.2 Le système GOLEM 107

4.4 Extension du système RuLe-Clust à la logique du premier ordre 108
4.4.1 Apprentissage disjonctif d'un concept 108
4.4.2 L'approche par décomposition 109
4.4.3 Exemple 1 : sous-concepts non-disjoints I 110
4.4.4 Exemple 2 : description numérique 111
4.5 Mesure de similarité 113
4.5.1 Langage infini 114
4.5.2 Expérimentations 117
4.6 Vue générale de la méthode 122
4.6.1 Apprentissage d'une clause 123
4.6.2 Complexité de la méthode globale 123
4.6.3 Expérimentations 124
4.7 Conclusion 126

5 Classification de données textuelles pour la Recherche d'Information 127
5.1 Introduction à la Recherche d'Information 128
5.2 Mesure de similarité et données textuelles 129
5.2.1 Évaluer la similarité entre les mots 129
5.2.2 Les mesures basées sur les cooccurrences 129
5.2.3 Les mesure basées sur des distributions 130
5.2.4 Les mesures utilisant les connaissances extérieures 130
5.2.5 Un nouvel indice de proximité basé sur le Web 131
5.2.6 Expérimentations sur l'indice de proximité contextuelle 134
5.3 Classification de mots 137
5.3.1 Travaux antérieurs 137
5.3.2 PoBOC : un algorithme approprié ? 138
5.3.3 Expérimentation 139
5.4 Regroupement de mots pour la classification de documents 143
5.4.1 Introduction à la classification de documents 143
5.4.2 Classification supervisée de documents 143
5.4.3 Réduction de la dimension de l'espace de description des documents 146
5.4.4 Le classifieur naïf de Bayes pour la classification de documents . . . 149
5.4.5 Regroupement distributionnel agglomératif 150
5.4.6 Partitionnement par optimisation d'une fonction de qualité, issue de la théorie de l'information 152
5.4.7 Réduction de l'espace par pseudo-partitionnement du vocabulaire . . 152
5.4.8 Étude expérimentale 158
5.5 Étude prospective : application à l'analyse des genres 161

Conclusion 165

A Mesures d'évaluation des schémas de clustering 169

B Évaluation du nombre de clusters 173

C Mesures de qualité d'une règle de classification 175

D Expertise de quelques groupes de documents **179**
D.1 Classification basée sur l'ensemble des 130 descripteurs 179
D.2 Étude des singletons 180
D.3 Classifications basées sur les autres ensembles de descripteurs 181

Table des figures

1.1 Exemples de clusters convexes (gauche) et concaves (droite), dans $\mathbb{R}^2$. . . . 10
1.2 Les différentes étapes du processus de clustering. 12
1.3 Exemple de dendrogramme. 21
1.4 Algorithme divisif hiérarchique DIANA. 22
1.5 Algorithme agglomératif hiérarchique SAHN. 23
1.6 Exemple de pyramide. 24
1.7 Algorithme de construction d'une pyramide. 25
1.8 Exemples de croisements (gauche et centre) et d'inversions (droite) dans les pyramides. 25
1.9 Algorithme divisif des k-moyennes flou hiérarchique. 26
1.10 Exemple de dendrogramme flou. 27
1.11 Exemple de Hiérarchie "IS A". 27
1.12 Algorithme de partitionnement k-moyennes. 28
1.13 Algorithme de partitionnement k-moyennes flou. 30
1.14 Algorithme de partitionnement k-médoïdes flou. 31
1.15 L'algorithme EM. 33
1.16 Structure hiérarchique 2D générée par STING. 35
1.17 L'algorithme WaveCluster. 37
1.18 L'algorithme DBSCAN. 40
1.19 Exemple d'une hiérarchie de concepts générée par COBWEB. 42
1.20 L'algorithme COBWEB. 43
1.21 Fonctions auxiliaires de COBWEB. 44

2.1 Exemples de pôles. 55
2.2 L'algorithme de clustering PoBOC. 57
2.3 Contrainte sur l'existence d'une arête entre deux sommets dans $G(X, V, d)$. 58
2.4 Organisation des objets de X dans $(\mathbb{R}^2, L_2)$. 59
2.5 Graphe de dissimilarité de X. 59
2.6 Heuristique de choix des objets de départ. 61
2.7 Exemple d'extraction de cliques quasi-identiques. 62
2.8 La procédure de construction d'un pôle. 63
2.9 Pôles obtenus sur X. 64
2.10 La procédure de multi-affectations d'un objet aux pôles. 65
2.11 Exemple de multi-affectations. 66

2.12 Schémas de clustering final. 66
2.13 Processus global de PoBOC. 67
2.14 Observations des pôles constitués par PoBOC sur la base de données Iris selon les coordonnées : x, y, z (a), x, y, t (b), x, z, t (c) et y, z, t (d). 70
2.15 Évaluation du nombre de clusters sur la base Iris. 73

3.1 Exemple d'un arbre de décision. 82
3.2 Un exemple de choix de sélecteur. 86
3.3 Exemple de décomposition d'une classe. 87
3.4 Exemple de partitionnement disjoint conduisant à des règles non-disjointes. 89
3.5 Exemple de partitionnement disjoint conduisant à des règles disjointes. . . . 89
3.6 Exemple de partitionnement non-disjoint conduisant à des règles non-disjointes. 90
3.7 Procédure principale du système RuLe-Clust. 91
3.8 Algorithme de couverture d'une classe par clustering. 91
3.9 Test d'existence d'une règle unique. 92
3.10 Heuristique de construction d'une règle. 93
3.11 Processus de décomposition pour l'algorithme des k-moyennes. 94

4.1 L'algorithme d'apprentissage de règles logiques FOIL. 106
4.2 Algorithme de construction d'une règle par généralisation : GOLEM. 107
4.3 Exemples décrits sur deux attributs numériques. 112
4.4 Matrice de similarité avec valeurs élevées grisées. 118
4.5 Matrices de similarité avec valeurs élevées grisées (variation du seuil). . . . 118
4.6 Performance du classifieur 1-PPV, pour des langages de tailles différentes. . 119
4.7 Performance du classifieur 1-PPV, pour différentes complexités de clauses. . 120
4.8 Performance du classifieur 1-PPV, pour différentes pondérations des domaines. 121
4.9 Vue générale de l'algorithme d'apprentissage disjonctif par décomposition. . 122
4.10 base de connaissance (à gauche) et représentation graphique (à droite). . . . 125

5.1 Matrices de proximité obtenues sur les 38 mots-clés : (a) indices de proximité basés sur les cooccurrences, (b) indices de proximité contextuelle. 136
5.2 Organisation hiérarchique du pseudo-partitionnement obtenu par PoBOC sur la base des 38 mots-clés. 140
5.3 Évaluation du nombre de clusters sur les 38 mots-clés. 141
5.4 Algorithme de clustering pour la méthode ADC. 151
5.5 Algorithme de clustering pour la méthode ITDC. 153
5.6 Algorithme de clustering pour la méthode DDOC. 156
5.7 Procédure de multi-affectations utilisée dans la méthode DDOC. 157
5.8 Évaluation de l'influence de la proportion d'intersections entre clusters, sur la performance du classifieur. Corpus *Newsgroup* (gauche) et *Reuters* (droite). 159
5.9 Évaluation de la méthode DDOC, comparativement aux méthodes ADC et ITDC sur les deux corpus : *Newsgroup* (gauche) et *Reuters* (droite). 161

B.1 Zoology. 173
B.2 Wine. 174
B.3 Soybean. 174
B.4 Annealing. 174

Liste des tableaux

1.1 Propriétés mathématiques des différentes mesures de dissimilarité. 16

2.1 Analyse des clusters obtenus par PoBOC sur la base de données Iris. 71
2.2 Table de contingence sur Iris. 71
2.3 Évaluation du schéma par des indices externes. 72
2.4 Évaluation du schéma par des indices relatifs. 74
2.5 Recherche du nombre optimal de clusters sur les bases : Zoology, Wine, Soybean et Annealing. 75
2.6 Évaluation des schémas par des indices externes. 76
2.7 Évaluation des schémas par des indices relatifs. 77

3.1 Présentation des ensembles de données. 96
3.2 Évaluation du système RuLe-Clust. 97
3.3 Comparaison de différentes méthodes de clustering. 98
3.4 Influence des intersections entre clusters. 99
3.5 Comparaison de différents classifieurs. 100

5.1 Nombre de pages Web retournées pour chaque requête de un ou deux mots. 132
5.2 Information Mutuelle pour chaque couple de mots. 132
5.3 Les 11 mots-clés du domaine *Ressources du Langage*. 134
5.4 Les 13 mots-clés du domaine *Web Mondial*. 135
5.5 Les 14 mots-clés du domaine *Intelligence Artificielle*. 135
5.6 Évaluation quantitative des schémas obtenus par différentes méthodes de clustering. 142
5.7 Tables de contingence pour la classe c_i (gauche) et globale (droite). 146

C.1 Table de contingence et notations. 176
C.2 Mesures d'évaluation de la performance d'une règle. 177

Introduction

La classification : passé/présent

Tout ce qui nous entoure, qu'il s'agisse de choses physiques ou abstraites, nous apparaît de façon organisée. Lorsque nous voyons un animal, nous le désignons systématiquement par l'espèce à laquelle il appartient. Un sentiment sera également catégorisé, on parlera par exemple d'un sentiment de tristesse, de joie ou de peur. Pourtant, chacun éprouve la peur différemment et les chiens ne sont pas tous identiques non plus. Il s'agit, en fait, d'un phénomène, conscient ou non, mais naturel et indispensable, de simplification pour mieux comprendre et mieux communiquer.

L'exemple que nous venons d'évoquer n'est pas anodin. En effet on attribue les premières recherches théoriques sur la classification, aux besoins exprimés par les biologistes, de spécifier des classes ou espèces animales [86]. Cette classification était censée permettre, étant donnée une description (simple) d'un animal, de l' "étiqueter" par tel ou tel nom d'espèce. Le problème s'est avéré plus complexe qu'il n'y paraissait...

Les applications se sont multipliées, chacune apportant son lot de problèmes, plus complexes les uns que les autres. Après les biologistes, ce sont les historiens, les médecins, les sociologues ou les didacticiens qui ont manifesté le besoin de classer leurs reliques, leurs patients ou leurs élèves. Aujourd'hui, en nous intéressant aux problèmes issus des technologies de l'information et de la communication, nous sommes amenés à traiter des données très complexes, par leur nature, leur taille, leur provenance, leur diversité. Ces données peuvent être, par exemple, du texte, des images, des séquences vidéo, des bandes soonores et tout support combinant ces différents types de données.

L'envie d'organiser pour simplifier a progressivement évolué vers l'ambition de classer (ou classifier) pour comprendre et, pourquoi pas, pour prédire. Cette évolution a conduit à dégager deux stratégies de classification : supervisée et non-supervisée.

La notion de prédiction fait référence à une stratégie particulière d'Apprentissage Automatique (AA), appelée apprentissage supervisé. Ce domaine d'étude à part entière, peut être considéré comme une sous-thématique de l'Intelligence Artificielle (IA). De façon synthétique, l'apprentissage supervisé consiste à faire émerger d'un ensemble de données d'entraînement pré-classifiées, les caractéristiques nécessaires et suffisantes pour permettre de classer correctement une nouvelle donnée. Dans ce type d'approche, les classes sont connues à l'avance, cette connaissance est utilisée dans le processus d'apprentissage.

La classification non-supervisée correspond davantage au problème initial d'organisation, mentionné ci-dessus, et puise ses fondements théoriques en Analyse de Données (AD).

En fonction de l'application, la tâche de classification non-supervisée a pour but d'organiser un ensemble d'objets sans connaissance *a priori* d'une classification, afin de simplifier, ou structurer les données par l'extraction ou la reconnaissance des "formes" récurrentes dans ces données. On parle, en apprentissage, du problème de la Reconnaissance de Formes (RF) faisant appel à des méthodes de regroupement (en anglais, *clustering*).

Sujet de la thèse

Dans un contexte supervisé, la classification est souvent formalisée par un problème, ou un ensemble de problèmes, de décision(s) binaire(s)[1]. Étant donnés un exemple e et une classe c, une fonction cible apprise permet de décider si e appartient ou n'appartient pas à la classe c. Il est alors naturel d'imaginer qu'un exemple e puisse appartenir à plusieurs classes. L'exemple des données textuelles (textes, mots, etc.) illustre parfaitement ce phénomène, une telle donnée étant rarement associée à une classe thématique exclusivement.

L'aspect "multi-classes" est donc rencontré et traité couramment dans les approches supervisées. Aussi naturel qu'il puisse être, il est pourtant ignoré dans la majorité des systèmes d'apprentissage non-supervisé, limitant de ce fait la représentativité des classes. Une explication de cette limitation, la principale à notre avis, est que l'espace de recherche (ensemble de tous les schémas de classification possibles, étant donné un ensemble de données) devient très important lorsqu'on autorise chaque objet à appartenir à plusieurs classes.

Les méthodes floues pourraient être considérées comme "la" solution à ce problème. En effet, les schémas flous, appris par les algorithmes de regroupement flou, effectuent en quelque sorte, un partage de chaque objet sur les différentes classes. Néanmoins, ce type d'approche souffre d'un manque de simplicité ; les schémas sont difficiles à visualiser, et ne correspondent pas aux attentes d'experts qui chercheraient par exemple à découvrir les relations qui existent ou non entre des objets.

Quelques propositions ont été faites pour modifier les algorithmes existants dans le but de restreindre les schémas de classification flous ou au contraire d'étendre les schémas disjoints afin d'obtenir des groupes d'objets avec intersections [46]. Malheureusement ces adaptations sont soit spécifiques à un domaine d'application (aux traitement des données textuelles par exemple), soit encore trop restrictives sur les propriétés des schémas générés (limitation de la taille des intersections, du nombre de groupes pouvant partager un même objet, etc.).

Dans cette thèse, nous postulons que *"la construction d'un schéma constitué de groupes non-disjoints, dans un processus d'apprentissage non-supervisé, offre une structuration pertinente d'un ensemble de données."*

La confirmation de cette hypothèse passe tout d'abord par la proposition d'une méthode de regroupement adaptée spécifiquement à la construction de groupes non-disjoints d'objets. Différents critères doivent être étudiés afin que cet algorithme puisse être utilisé dans un large éventail d'applications et réponde aux besoins de la tâche de classification. Parmi ces critères, on s'attache en particulier à la possibilité de découvrir automatiquement un nombre approprié de groupes, de tailles et densités variées, à partir de données

[1] Certaines méthodes d'apprentissage supervisé proposent une formalisation discrète du problème, en associant un score (une probabilité par exemple) à chaque décision. Cependant, la décision finale est souvent obtenue en utilisant un seuil et en se ramenant ainsi à une décision binaire.

caractérisées soit par un ensemble de descripteurs, soit par les relations des objets entre eux, uniquement.

Le développement d'une nouvelle méthode s'accompagne naturellement d'une phase d'évaluation ; or le problème de l'évaluation se pose de façon récurrente en apprentissage non-supervisé. L'utilisation des mesures traditionnelles de qualité est insuffisante, ce qui nous amène à envisager d'autres alternatives pour mesurer l'intérêt de notre approche.

Deux applications très différentes sont étudiées. Tout d'abord, dans un cadre général d'apprentissage de règles, nous utilisons l'algorithme de regroupement pour structurer les données d'un concept cible avant d'entreprendre le processus de construction de règle(s)[2]. Cette application présente deux intérêts majeurs : elle offre d'une part une évaluation objective de l'algorithme, et d'autre part, cette phase d'organisation préalable des données peut faire l'objet d'une étude porteuse dans le domaine de l'apprentissage de règles propositionnelles ou logiques.

Dans un deuxième temps, nous abordons le problème de l'organisation des données textuelles dans le cadre de la Recherche d'Information (RI). Cette application est incontournable pour notre approche, car elle constitue l'une des principales motivations des schémas de groupes non-disjoints.

Organisation du mémoire

Ce mémoire comporte 5 chapitres :

Le premier chapitre, intitulé *"État de l'art du clustering"*, dresse un panorama des différentes approches proposées pour regrouper des données. Nous présentons en particulier les méthodes hiérarchiques, de partitionnement, basées sur un mélange de lois de probabilité, sur un découpage de l'espace de représentation ou sur la notion de voisinages (densités). Nous essayons d'envisager les différents types de schémas de classification possibles à établir, par le biais de ces approches (groupes disjoints, non-disjoints et flous).

La méthode de regroupement que nous proposons pour construire un ensemble de groupes non-disjoints est présentée dans le second chapitre. L'algorithme PoBOC (*Pole-Based Overlapping Clustering*) est alors présenté de façon formelle et intuitive. Une première série d'évaluations est effectuée, sur des jeux de données traditionnels, en utilisant des mesures de qualité définies dans le chapitre précédent.

L'utilisation de l'algorithme PoBOC dans le cadre de l'apprentissage de règles est à l'origine des deux chapitres suivants. Le chapitre 3 présente un système d' *"apprentissage de règles de classification par regroupement d'objets"*. Les règles sont apprises dans un formalisme propositionnel (attribut/valeur). Deux objectifs sont alors visés : d'une part montrer que l'algorithme PoBOC, et en particulier le type de schéma qu'il génère, est performant pour cette tâche d'apprentissage, d'autre part qu'il est pertinent de prévoir cette étape d'organisation des données avant le processus de construction de règles.

Cette dernière application est étendue, dans le chapitre 4, à l'apprentissage de règles logiques pour la définition de concepts disjonctifs. Étant donné un langage de représentation, on montre que la décomposition des concepts par un algorithme de regroupement, et en particulier par l'algorithme PoBOC, permet de guider la stratégie de recherche d'hypothèses satisfaisantes pour définir un concept cible.

[2]Ce système global se situe dans un cadre supervisé, et l'une des étapes est effectuée par un processus non-supervisé.

Enfin, le dernier chapitre se concentre sur l'utilisation de notre approche dans le cadre de la *"classification de données textuelles pour la recherche d'information"*. Nous montrons d'abord sur des exemples simples à analyser, que l'organisation de mots en groupes non-disjoints offre une structuration précise et exploitable des données. Puis, nous utilisons ce résultat pour motiver l'indexation de documents par des groupes non-disjoints de mots, dans un processus global de classification de documents. Enfin, nous dépassons le cadre de l'évaluation de notre approche, dans une étude prospective sur l'interaction genre/domaine en recherche d'information.

1

État de l'art du Clustering

Sommaire

1.1 Clustering et apprentissage 7
1.2 Cadre formel du clustering 7
1.2.1 Partitions, pseudo-partitions et partitions floues 8
1.2.2 Hiérarchies et pseudo-hiérarchies 8
1.2.3 Centroïdes et médoïdes 9
1.2.4 Concavité vs. convexité d'un cluster 10
1.2.5 Les outliers 10
1.2.6 Autres notions utiles 11
1.3 Les trois principales étapes du clustering 11
1.3.1 La préparation des données 12
1.3.2 Le choix de l'algorithme 13
1.3.3 L'exploitation des clusters 15
1.4 La similarité 15
1.4.1 Propriétés formelles de base 15
1.4.2 Similarité et variables numériques 17
1.4.3 Similarité et variables symboliques 17
1.5 Clustering "dur", clustering avec recouvrements et clustering "flou" 19
1.6 Les différentes méthodes de clustering 20
1.6.1 Le clustering hiérarchique 20
1.6.2 Le clustering par partitionnement 27
1.6.3 Le clustering par mélange de densités de probabilités 32
1.6.4 Le clustering par grilles 34
1.6.5 Le clustering par densités 38
1.6.6 Le clustering conceptuel 41
1.6.7 Quelques autres approches de clustering 45
1.7 Techniques d'évaluation du clustering 46
1.7.1 Critères externes 48

1.7.2 Critères internes 48
1.7.3 Critères relatifs 49
1.7.4 Critères d'évaluation pour clustering flou 50
1.7.5 Adaptation au clustering avec recouvrements 50
1.8 Conclusion sur le clustering 51

1.1 Clustering et apprentissage

Le clustering (ou regroupement) est un sujet de recherche en apprentissage émanant d'une problématique plus générale, à savoir la classification. On distingue la classification supervisée et non supervisée. Dans le premier cas, il s'agit d'apprendre à classer un nouvel individu parmi un ensemble de classes prédéfinies, à partir de données d'entraînement (couples *(individu,classe)*). Issue des statistiques, et plus précisément de l'Analyse De Données (ADD), la classification non-supervisée, comme son nom l'indique, consiste à apprendre sans superviseur. A partir d'une population, il s'agit d'extraire des classes ou groupes d'individus présentant des caractéristiques communes, le nombre et la définition des classes n'étant pas donnés *a priori*.

La classification non-supervisée fut, dans un premier temps, utilisée dans les domaines d'application suivants :

- la médecine [167] : découverte de classes de patients présentant des caractéristiques physiologiques communes,
- le traitement de la parole [151] : construction de systèmes de reconnaissance de l'orateur,
- l'archéologie [83] : regroupement d'objets datant de l'âge de fer à partir de leur description,
- l'éducation [99] : structuration d'une classe d'ouvriers dans le domaine de l'industrie du téléphone, à partir de leurs besoins communs de formation.

Des méthodes "simples" datant des années 1950-60 et issues des statistiques, telles que les approches par construction de taxonomies (par exemple [94]), offraient aux experts la possibilité d'analyser et d'explorer leurs données.

Progressivement, d'autres applications sont apparues (traitement d'images, de données textuelles, etc.) imposant alors de nouvelles contraintes de performances, d'utilisations ou de flexibilités pour ces méthodes. Par ailleurs, le développement d'outils informatiques puissants a permis d'envisager de nouvelles pistes de recherche dans le domaine du clustering. Depuis les années 1990, on peut considérer que le clustering constitue un domaine d'étude incontournable en apprentissage. Ces techniques occupent en effet une large place dans les processus de pré-traitement, structuration, organisation des données et d'Extraction de Connaissances dans les Données (ECD).

Nous décrivons un cadre formel pour le problème du clustering ainsi que la définition des notions de bases utiles pour la suite de l'étude.

1.2 Cadre formel du clustering

Dans la suite, nous utiliserons le terme "regroupement" ou son équivalent anglais "clustering", pour désigner le processus de classification non-supervisée. De même, le résultat d'un tel processus sera appelé parfois "schéma de clustering" ou plus simplement "classification".

On considère un ensemble de n objets $X = \{x_1, \ldots, x_n\}$, ainsi qu'une matrice de dissimilarité D sur cet ensemble, telle que $d(x_i, x_j)$ représente la dissimilarité entre les deux objets x_i et x_j. La matrice D est de taille $n \times n$ et à valeurs dans $[0, 1]$. Lorsque les données sont uniquement décrites par une telle matrice, on parle parfois de "données relationnelles".

Nous verrons, par la suite, que la plupart des méthodes de classification non-supervisées utilisent une description vectorielle des données. On parle alors de "données objets" et on considère un ensemble fini $\mathcal{V} = \{v_1, \ldots, v_p\}$ de variables descriptives, telles que $v_j(x_i)$ désigne la valeur de l'objet $x_i \in X$ pour la variable $v_j \in \mathcal{V}$.

La tâche de clustering permet de générer un ensemble de t clusters $\mathcal{C} = \{C_1, \ldots, C_t\}$ tel que chaque cluster C_a est un sous-ensemble de X ($C_a \subset X$) et l'union des clusters couvre l'ensemble des objets de départ ($\bigcup_{a=1}^{t} C_a = X$).

1.2.1 Partitions, pseudo-partitions et partitions floues

Définition 1.1. *$\mathcal{C}$ est une* **partition** *de X si et seulement si $\mathcal{C}$ vérifie les propriétés suivantes :*

i) $C_a \subset X$ *pour tout* $C_a \in \mathcal{C}$,
ii) $\bigcup_{a=1}^{t} C_a = X$,
iii) $C_a \cap C_b = \emptyset$ *pour* (a, b) *tel que* $a \neq b$.

La propriété *iii)* exprime le fait que les clusters constitués sont disjoints, chaque objet de X ne peut donc appartenir qu'à un seul cluster de $\mathcal{C}$. Une telle partition sera également appelée "partition stricte".

Définition 1.2. *$\mathcal{C}$ est une* **pseudo-partition** *de X si et seulement si $\mathcal{C}$ vérifie les propriétés suivantes :*

i) $C_a \subset X$ *pour tout* $C_a \in \mathcal{C}$,
ii) $\bigcup_{a=1}^{t} C_a = X$,
iii) $C_a \subseteq C_b$ *ssi* $a = b$.

Dans le cas d'une pseudo-partition, les intersections entre clusters ne sont pas nécessairement vides. Cependant, la propriété *iii)* interdit qu'un cluster soit inclus dans un autre.

Dans les deux définitions précédentes, chaque objet x_i appartient ou non à un cluster C_a donné. On peut alors formaliser le processus de construction d'une partition ou d'une pseudo-partition par la donnée de t fonctions à valeurs binaires :

$$u_a \ : \ X \rightarrow \{0, 1\}, \ a = 1 \ldots t \ \textit{avec} \ u_a(x_i) = \begin{cases} 1 & \textit{si } x_i \in C_a \\ 0 & \textit{sinon.} \end{cases}$$

Cette formalisation peut être généralisée au cas de fonctions à valeurs réelles. Dans ce cas, les partitions générées sont dites "floues".

Définition 1.3. *Une* **partition floue** *de X, notée $\mathcal{C} = \{C_1, \ldots, C_t\}$, est définie par la donnée de t fonctions*

$$u_a \ : \ X \rightarrow [0, 1], \ a = 1 \ldots t$$

$u_a(x_i)$ représente alors le degré d'appartenance de l'objet x_i au cluster C_a.

1.2.2 Hiérarchies et pseudo-hiérarchies

Certaines méthodes de clustering conduisent à un arbre hiérarchique aussi appelé dendrogramme. De même que l'on distingue les partitions strictes des pseudo-partitions, cette même distinction est faite entre hiérarchies et pseudo-hiérarchies.

Définition 1.4. *Soit P un ensemble de parties non vides sur X, P est une* **hiérarchie** *si les propriétés suivantes sont vérifiées :*

i) $X \in P$,
ii) *pour tout* $x_i \in X,\ \{x_i\} \in P$,
iii) *pour tout* $h, h' \in P,\ h \cap h' \in \{\emptyset, h, h'\}$,
iv) *pour tout* $h \in P,\ \cup\{h' \in P\ :\ h' \subset h\} \in \{h, \emptyset\}$.

Les propriétés *i)* et *ii)* expriment le fait que la racine de l'arbre est constituée de l'ensemble X et que les feuilles de l'arbre correspondent aux singletons $\{x_1\}, \ldots, \{x_n\}$. Les propriétés *iii)* et *iv)* assurent que deux clusters ne s'intersectent que si l'un est inclus dans l'autre et que chaque cluster contient tous ses successeurs ("fils") et est contenu dans son unique cluster prédécesseur ("père").

Définition 1.5. *Soit P un ensemble de parties non vides sur X, P est une* **pseudo-hiérarchie** *si les propriétés suivantes sont vérifiées :*

i) $X \in P$,
ii) *pour tout* $x_i \in X,\ \{x_i\} \in P$,
iii) *pour tout* $h, h' \in P,\ h \cap h' = \emptyset$ *ou* $h \cap h' \in P$,
iv) *il existe un ordre (total)* θ *sur* X *compatible avec* P.

Définition 1.6. *Un ordre θ est* **compatible** *avec un ensemble P de parties de X, si tout élément de $h \in P$ est connexe selon θ.*

Définition 1.7. *Une partie h est* **connexe** *selon θ si x et y étant les bornes (i.e. le plus petit et le plus grand élément) de h selon θ, on a la condition :*

$$\{z \text{ compris entre } x \text{ et } y \text{ selon } \theta\} \Leftrightarrow \{z \in h\}$$

Par ces définitions, une pseudo-hiérarchie (aussi appelée pyramide [46]) est telle que chaque cluster peut avoir plusieurs prédécesseurs. De plus, la propriété *iv)* concernant l'existence d'un ordre θ sur X, permet la visualisation d'une telle pyramide.

La notion de hiérarchie floue n'est pas formellement définie. Ce type d'organisation est en effet très peu utilisée en apprentissage (cf. section 1.6.1).

1.2.3 Centroïdes et médoïdes

De nombreux algorithmes de clustering assimilent chaque cluster à un point, ceci pour des raisons pratiques de complexité notamment. Ce "point", censé être représentatif du cluster peut être un centroïde ou un médoïde.

Définition 1.8. *Le* **centroïde** *x^* d'un cluster C_a est le point défini dans $\mathcal{V}$ par :*

$$\forall j = 1, \ldots, p, \quad v_j(x^*) = \frac{1}{|C_a|} \sum_{x_i \in C_a} v_j(x_i)$$

Dans le cas où toutes les variables descriptives $v_1, \ldots, v_p$ sont quantitatives (continues ou discrètes), le centroïde est défini, sur chaque composante, par la valeur moyenne des objets du cluster pour cette même composante. En ce sens, le centroïde correspond au centre de gravité du cluster. Le centroïde d'un cluster ne fait généralement pas partie des objets constituant ce cluster.

En revanche, lorsque certaines variables descriptives sont de nature qualitative (ex. couleur, forme, etc.), la définition précédente n'a plus de sens puisque les opérateurs classiques (addition, division, etc.) ne sont pas applicables sur ce type de variables. On cherche alors l'objet le plus représentatif du cluster que l'on appelle aussi parfois "prototype" ou plus formellement "médoïde".

Définition 1.9. *Soient un cluster C_a d'objets définis sur $\mathcal{V}$, et d une mesure de dissimilarité sur $\mathcal{V}$, le* **médoïde** *du cluster C_a est l'objet $x^* \in C_a$ tel que :*

$$x^* = \underset{x_i \in C_a}{\arg\min} \frac{1}{|C_a|} \sum_{x_j \in C_a} d(x_i, x_j)$$

Par cette définition, le médoïde d'un cluster est l'objet du cluster tel que la dissimilarité moyenne de tous les objets du cluster avec le médoïde est minimale. Inversement, le médoïde d'un cluster est l'objet en moyenne le plus similaire aux autres.

1.2.4 Concavité vs. convexité d'un cluster

On est parfois amené à s'intéresser à la "forme" des clusters obtenus par un algorithme de clustering. Notons que cette notion de "forme" est difficile à définir formellement et pourrait laisser penser que les objets doivent systématiquement être représentés dans un espace. Par exemple, dans un espace à deux dimensions, les formes convexe et concave peuvent être illustrées par la figure 1.1.

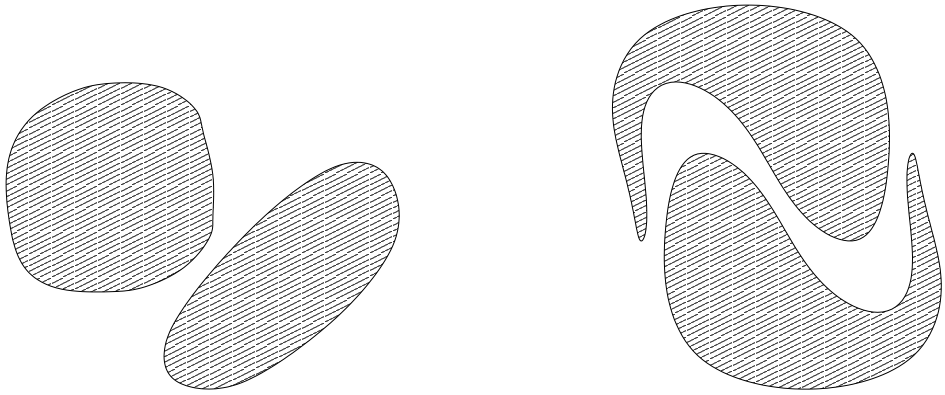


FIG. 1.1 – Exemples de clusters convexes (gauche) et concaves (droite), dans $\mathbb{R}^2$.

On peut néanmoins parler de convexité ou concavité des clusters, pour des objets sur lesquels on ne connaît pas d'espace de représentation *a priori*. De façon générale, un cluster est convexe lorsque les objets qui le composent sont organisés autour d'un centre (centroïde ou médoïde). On remarquera dans la suite, que l'algorithme des k-moyennes, par exemple, construit des clusters convexes, par affectation des objets à un centre. A l'inverse, un algorithme tel que l'algorithme agglomératif hiérarchique du simple lien, agglomère les objets "de proche en proche" et peut renvoyer des clusters concaves.

1.2.5 Les outliers

Les *outliers* sont des observations peu fréquentes qui sortent de la norme (observations atypiques). Une autre façon de les définir est de considérer que les outliers sont des observations qui dévient tellement des autres observations que l'on peut soupçonner qu'elles

aient été générées par un processus différent [80]. Il n'existe pas de définition formelle d'un outlier telle que Hawkins le conçoit, cependant la notion de $DB(m, \delta)$-*Outlier*[1] - consistante avec la définition de Hawkins - est souvent utilisée [100].

Définition 1.10. *Un objet $x^* \in X$ est un $DB(m, \delta)$-outlier s'il existe un sous-ensemble X' de X, constitué d'au moins m objets $x'_1, \ldots, x'_m$, tel que $\forall x'_i \in X'$, $d(x^*, x'_i) > \delta$, où d est une mesure de dissimilarité définie sur X.*

Comme nous le verrons par la suite, la gestion de ce type de données est difficile mais très importante puisque les outliers peuvent introduire un biais dans le processus d'analyse.

1.2.6 Autres notions utiles

Nous définissons enfin plusieurs notions permettant de caractériser les clusters. Notons que ces critères se retrouvent dans la plupart des méthodes de clustering qui consistent à optimiser une fonction de qualité. Dans les définitions suivantes, nous considérerons un cluster C_a composé d'objets de X ; x^*_a correspondant au centre du cluster (centroïde ou médoïde) ; et d étant une mesure de dissimilarité (ou distance) sur X.

Définition 1.11. *Le* **rayon** *d'un cluster correspond à la plus grande distance du centre à un autre élément*

$$rayon(C_a) = \max_{\{x_i \in C_a\}} d(x_i, x^*_a)$$

Définition 1.12. *Le* **diamètre** *d'un cluster est égal à la plus grande distance entre deux éléments de ce cluster*

$$diam(C_a) = \max_{\{x_i, x_j \in C_a\}} d(x_i, x_j)$$

Définition 1.13. *L'***inertie** *d'un cluster (inertie intra-cluster) correspond à la somme des carrés des distances au centre*

$$I_{intra}(C_a) = \sum_{x_i \in C_a} d(x_i, x^*_a)^2$$

Définition 1.14. *Étant donné un schéma de clustering $\mathcal{C} = \{C_1, \ldots, C_t\}$, l'***inertie** *inter-clusters correspond à la somme des carrés des distances entre les centres des clusters*

$$I_{inter}(\mathcal{C}) = \sum_{i=2}^{t} \sum_{j<i} d(x^*_i, x^*_j)^2$$

Définition 1.15. *La* **variance** *d'un cluster est égale à la moyenne des carrés des distances au centre*

$$V(C_a) = \frac{1}{|C_a|} \sum_{x_i \in C_a} d(x_i, x^*_a)^2$$

1.3 Les trois principales étapes du clustering

Le processus de clustering se divise en trois étapes majeures (figure 1.2) : (1) la préparation des données, (2) l'algorithme de clustering et (3) l'exploitation des résultats de l'algorithme. Nous discutons ici des problématiques générales liées à chacune de ces étapes.

[1] Pour *Distance-Based outlier.*

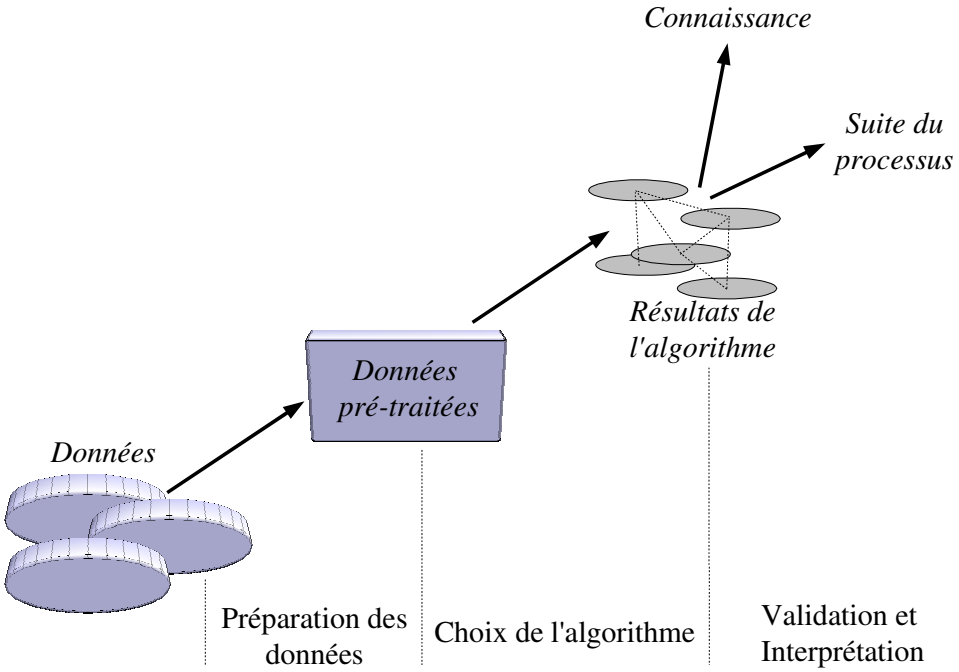


FIG. 1.2 – Les différentes étapes du processus de clustering.

1.3.1 La préparation des données

Variables et sélections

Les objets sont décrits par des variables $\{v_j\}_{j=1...p}$, aussi appelées attributs, descripteurs ou traits. Ces variables sont de différentes natures :

- variables *quantitatives* : continues (e.g. la taille d'une personne), discrètes (e.g. le nombre de personnes) ou sous forme d'intervalles (e.g. la période de vie d'une personne),
- variables *qualitatives* : non-ordonnées (e.g. la "couleur" des cheveux) ou ordonnées (e.g. la taille : "petit", "moyen", "grand", etc.),
- variables *structurées* : par exemple la forme d'un objet (polygone, parallélogramme, rectangle, ovale, cercle, etc.)

L'étape de préparation consiste à sélectionner et/ou pondérer ces variables, voire à créer de nouvelles variables, afin de mieux discriminer entre eux les objets à traiter. En effet les variables ne sont pas nécessairement toutes pertinentes : certaines peuvent être redondantes et d'autres non-pertinantes pour la tâche ciblée. Ce problème de sélection de variables a été largement étudié en classification supervisée mais reste très ouvert pour une approche non-supervisée.

Dans un cadre supervisé, chaque variable peut-être évaluée relativement à son pouvoir discriminant par rapport aux classes prédéfinies. Deux types de méthodes se dégagent : les méthodes "filtre" (*filter*) et les méthodes "enveloppe" (*wrapper*). Dans le premier cas il s'agit d'enlever les variables non pertinentes avant la phase d'apprentissage alors que les approches "enveloppe" utilisent de manière explicite le classifieur pour choisir un sous-ensemble de variables réellement discriminantes [116].

En revanche, peu de travaux concernent la sélection de variables dans une perspective non-supervisée, principalement parce que, sans les étiquettes de classe, il est difficile d'évaluer la pertinence d'une variable. Cette difficulté peut être contournée en effectuant une première étape de clustering à partir de l'ensemble des variables, puis de considérer chaque cluster comme une classe et de réitérer ce processus [106]. De fait, cette technique se place parmi les approches "enveloppe" puisqu'elle est fortement dépendante de l'algorithme de clustering et des paramètres utilisés (nombre de clusters, etc.). Récemment, des approches "filtres" ont été envisagées dans ce contexte non-supervisé ([127, 34]).

Distances et similarités

La plupart des algorithmes de clustering utilisent une mesure de proximité entre les objets à traiter. Cette notion de "proximité" est formalisée à l'aide d'une mesure (ou indice) de similarité, dissimilarité ou encore par une distance. Le choix ou la construction de cette mesure est déterminant pour la suite du processus. En effet, deux mesures différentes peuvent induire deux schémas de classification différents. Finalement, chaque domaine d'application possédant ses propres données, il possède également sa propre notion de "proximité" ; il faut concevoir alors une mesure différente pour chaque domaine d'application, permettant de retranscrire au mieux les différences (entre les objets) qui semblent importantes pour un problème donné.

On note par exemple que les données textuelles (mots, documents, etc.) font appel à des indices ou mesures de similarités basées sur des cooccurrences (coefficient de Dice [169], indice d'équivalence [133], etc.). En revanche, les données spatiales sont généralement étudiées dans des espaces métriques en utilisant des mesures de distances traditionnelles (distance Euclidienne, distance de Manhattan, cosinus, etc.). Cependant, ces mêmes mesures peuvent être réutilisées dans d'autres domaines d'applications à condition qu'elles respectent les contraintes (propriétés) liées aux domaines considérés. Ainsi, il n'est en général pas nécessaire de redéfinir une telle mesure dans tous les cas.

Pour être plus précis sur cette notion fondamentale, nous proposons une présentation formelle de la "similarité" ainsi que ses enjeux dans le cadre du clustering, en section 1.4.

1.3.2 Le choix de l'algorithme

Le choix de l'algorithme de clustering doit donner lieu à une analyse globale du problème : quelle est la nature (qualitative et quantitative) des données ? Quelle est la nature des clusters attendus (nombre, forme, densité, etc.) ? L'algorithme doit alors être choisi de manière à ce que ses caractéristiques répondent convenablement à ces deux dernières questions. Les critères de décision peuvent être : la quantité de données à traiter, la nature de ces données, la forme des clusters souhaités ou encore le type de schéma attendu (pseudo-partition, partition stricte, dendrogramme, etc.).

La taille des données

La quantité d'objets à traiter est un premier facteur de décision. En effet, pour des données de très grande taille (par exemple en traitement d'images), les algorithmes de complexité plus que linéaires ($O(\alpha.n)$ sur le nombre n d'objets) sont quasiment prohibés. Ainsi des méthodes telles que l'algorithme des k-moyennes, proposé en 1967 par MacQueen [117] ou plus généralement la méthode des nuées dynamiques [44], étant de complexité linéaire, sont très souvent utilisés. En revanche, lorsque l'on souhaite organiser quelques milliers, voire quelques centaines d'objets, il est possible d'avoir recours à des méthodes plus complexes et nécessitant un temps de traitement plus important (méthodes hiérarchiques ou de partitionnement plus élaborées).

La nature des données

Comme nous l'avons précisé, beaucoup d'algorithmes de clustering s'appuient sur une matrice de similarité ou dissimilarité. Le plus souvent, cette matrice est obtenue à partir des descriptions des données. La nature de ces descriptions (variables qualitatives et/ou quantitatives), détermine alors le choix de la mesure de (dis)similarité utilisée.

Par ailleurs, on peut ne pas souhaiter traduire les données dans une telle matrice et conserver la table initiale des descriptions. Certaines méthodes, telles que le clustering conceptuel et notamment l'algorithme COBWEB [60] ou sa variante numérique CLASSIT [66], le permettent.

Quand bien même une matrice de similarité existe, certaines méthodes de clustering s'appuient sur la notion d'espace métrique. Pour l'algorithme des k-moyennes par exemple, cet espace métrique permet de définir de nouveaux objets (ici les centroïdes) absents de l'ensemble initial des données. Dans le cas où un tel espace n'est pas présent, le processus de clustering se base uniquement sur la mesure de similarité. Des variantes sont alors envisageables telles que l'algorithme des nuées dynamiques proposé par Diday en 1972 [44] ou plus simplement les algorithmes "k-médoïdes" ou PAM (*Partitioning Arround Medoids*) [96].

La forme des clusters

Certaines méthodes de clustering aboutissent à des clusters de formes spécifiques. Par exemple, les méthodes hiérarchiques ou de partitionnement qui consistent à déterminer des centroïdes/ médoïdes aboutissent le plus souvent à des clusters de forme convexe (*e.g.* des hyper-sphères). En revanche, des algorithmes tels que CHAMELEON [95] ou plus simplement l'algorithme agglomératif hiérarchique du simple lien [171], construisent des clusters de formes variées. La question est alors de savoir si il est raisonnable de formuler *a priori*, une hypothèse sur la forme des clusters.

Les variations de taille et densité ainsi que la gestion des "outliers" sont également à prendre en compte dans le choix de l'algorithme de clustering. On entend, par variation de taille, la capacité d'un algorithme à obtenir à la fois des clusters contenant beaucoup d'objets, et des clusters contenant peu voire très peu d'objets. De même, la prise en compte de la densité permet ou non d'obtenir des clusters contenant des objets plus ou moins "proches", au sens de la mesure de similarité établie. Enfin, la méthode de clustering peut être ou non sensible à la présence d'outliers.

Le type de résultats attendus

La sortie d'un algorithme de clustering peut être, par exemple, une partition (ou pseudo-partition), une fonction ou encore un dendrogramme (arbre hiérarchique). De même, chaque cluster obtenu peut être défini soit par l'ensemble des objets qui le composent, soit par une description relative aux variables initiales. On parle d'une définition en extension, dans le premier cas, et en intension, dans le second. Précisons que la plupart des approches proposent une classification à partir d'une définition en extension des clusters.

Une nouvelle fois, le choix de la méthode de clustering devra être fait en fonction du type de résultat souhaité et donc de l'exploitation envisagée de ce résultat.

1.3.3 L'exploitation des clusters

La tentation est grande, pour un non-spécialiste, de considérer comme "acquis" le résultat d'un processus de clustering. Autrement dit, les clusters obtenus ne sont généralement ni remis en cause ni évalués en terme de disposition relative, dispersion, orientation, séparation, densité ou stabilité [70]. Pourtant, il est sans aucun doute utile de distinguer les classes pertinentes obtenues, des autres. De même, cette étape d'analyse permet d'envisager le recours à une autre approche de clustering plus adaptée. Deux situations sont possibles : soit la tâche de clustering s'inscrit dans un traitement global d'apprentissage, soit les clusters générés par clustering constituent un résultat final.

Dans le premier cas, l'analyse des clusters obtenus (mesures statistiques de qualité) peut aider à orienter le traitement suivant. Une description des clusters n'est pas nécessaire dans cette situation.

En revanche, dans le cas où le clustering constitue à lui seul un processus global de découverte de classes, l'exploitation des clusters pour une application donnée passe par une description de ces derniers. Lorsque les objets se présentent sous la forme d'une matrice de (dis)similarité, il existe peu de méthodes pour décrire les classes (médoïdes, k objets représentatifs et mesures de cohésion). Lorsque les objets sont décrits par un ensemble de variables, on peut avoir recours à des méthodes de description des classes (descriptions conceptuelles).

1.4 La similarité

Comme nous l'avons déjà évoqué, la similarité est une notion utilisée dans la plupart des processus de clustering. Les quatre principaux domaines qui étudient la similarité sont : l'Analyse des Données (AD), la Reconnaissance des Formes (RF), l'Apprentissage Symbolique (AS) et les Sciences Cognitives (SC). Les trois tâches majeures utilisant la notion de similarité sont : la classification, l'identification et la caractérisation [15]. Nous nous intéressons dans ce travail, aux mesures de similarité/dissimilarité utilisées dans le cadre de la classification non-supervisée, une tâche commune à l'Analyse de Données et à la Reconnaissance des formes.

1.4.1 Propriétés formelles de base

Une "mesure" de similarité est une application symétrique s de $X \times X$ dans $\mathbb{R}^+$ telle que $s(x_i, x_i)$ est maximale et $s(x_i, x_j)$ est d'autant plus élevée que les descriptions des objets x_i et x_j sont similaires. De façon opposée, on peut définir une mesure de "dissimilarité",

dont la nature mathématique peut différer suivant les propriétés vérifiées par cette mesure. Les propriétés de minimalité, symétrie, identité, inégalité triangulaire et ultramétrique permettent de définir les notions d'indice de dissimilarité ou de distance, de distance simple ou ultramétrique et enfin d'écart simple ou ultramétrique. Nous définissons rapidement ces propriétés puis présentons chacune des mesures relativement aux propriétés qu'elles vérifient, dans le Tableau 1.1 .

Propriété 1.1. (Minimalité) *Une mesure de dissimilarité* $d: X \times X \rightarrow \mathbb{R}^+$ *vérifie la propriété de minimalité si et seulement si :*

$$\forall x_i \in X,\ d(x_i, x_i) = 0$$

Propriété 1.2. (Symétrie) *Une mesure de dissimilarité* $d: X \times X \rightarrow \mathbb{R}^+$ *est symétrique si et seulement si :*

$$\forall x_i, x_j \in X,\ d(x_i, x_j) = d(x_j, x_i)$$

Propriété 1.3. (Identité) *Une mesure de dissimilarité* $d: X \times X \rightarrow \mathbb{R}^+$ *vérifie la propriété d'identité si et seulement si :*

$$\forall x_i, x_j \in X,\ d(x_i, x_j) = 0 \Rightarrow x_i = x_j$$

Propriété 1.4. (Inégalité triangulaire) *Une mesure de dissimilarité* $d: X \times X \rightarrow \mathbb{R}^+$ *vérifie l'inégalité triangulaire si et seulement si :*

$$\forall x_i, x_j, x_k \in X,\ d(x_i, x_j) \leq d(x_i, x_k) + d(x_k, x_j)$$

Propriété 1.5. (Inégalité ultramétrique) *Une mesure de dissimilarité* $d: X \times X \rightarrow \mathbb{R}^+$ *vérifie l'inégalité ultramétrique si et seulement si :*

$$\forall x_i, x_j, x_k \in X,\ d(x_i, x_j) \leq \max\{d(x_i, x_k), d(x_k, x_j)\}$$

Type de mesure	Minimalité	Symétrie	Identité	Inégalité triangulaire	Inégalité ultramétrique
Indice de dissimilarité	X	X			
Indice de distance	X	X	X		
Distance	X	X	X	X	
Distance ultramétrique	X	X	X	X	X
Écart	X	X		X	
Écart ultramétrique	X	X			X

TAB. 1.1 – Propriétés mathématiques des différentes mesures de dissimilarité.

Le plus souvent, on utilise la notion de distance pour évaluer la dissimilarité entre deux objets. Le passage d'une distance à une mesure de similarité peut par exemple être effectué par l'égalité suivante :

$$\forall x_i, x_j \in X,\ s(x_i, x_j) = d_{max} - d(x_i, x_j)$$

où d_{max} correspond à la distance maximum séparant deux objets de X. Dans ce qui suit, nous présentons des mesures de (dis)similarités très utilisées en AD et RF, d'abord lorsque les données sont décrites par des variables numériques, puis lorsque toutes ou partie des variables sont de nature symboliques.

1.4.2 Similarité et variables numériques

La distance la plus connue, et sans aucun doute la plus utilisée, est la distance de Minkowski définie par

$$d(x_i, x_j) = (\sum_{k=1}^{p} |v_k(x_i) - v_k(x_j)|^l)^{1/l}$$

où $v_k(x_i)$ correspond à la valeur de l'objet x_i sur la variable v_k. Selon les valeurs prises par le paramètre l, on parle de distance Euclidienne ($l = 2$), de Manhattan ($l = 1$) ou de Chebychev ($l = \infty$). Une autre distance très utilisée, notamment dans des applications portant sur les données textuelles, est la distance du cosinus, correspondant au cosinus de l'angle θ formé par les deux vecteurs x_i et x_j :

$$d(x_i, x_j) = \cos(\theta) = \frac{x_i \cdot x_j}{||x_i||\ ||x_j||},$$

où "$\cdot$" désigne le produit scalaire et $||x_i||$ la norme de x_i ($\sqrt{\sum_k v_k(x_i)^2}$).

Les mesures précédentes supposent l'indépendance entre les variables descriptives (ou attributs) ce qui n'est bien sûr pas le cas en réalité pour la plupart des applications. La distance de Mahalanobis permet d'éviter cette hypothèse d'indépendance, elle se définit ainsi :

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^T S^{-1} (x_i - x_j)}$$

où S désigne la matrice de variance/covariance. Notons que pour $S = I$ (matrice identité), on se ramène alors à la distance Euclidienne. Enfin, la pondération des attributs est une étape importante dans la préparation des données, qui permet d'éviter de considérer davantage les attributs dont les valeurs sont fortement dispersées.

1.4.3 Similarité et variables symboliques

Dans le cas où les données sont décrites (totalement ou partiellement) par des variables symboliques, les mesures précédentes ne conviennent plus, puisque la différence entre deux modalités d'un même attribut n'a plus de sens. Une stratégie pour évaluer la similarité entre deux objets décrits de façon catégorielle est de redéfinir l'espace de description à l'aide d'attributs binaires puis d'effectuer un comptage des propriétés partagées ou non par les deux objets. Les indices de *Rand* [152] et de *Jaccard* [90] notés respectivement R et J procèdent de cette façon :

$$R(x_i, x_j) = \frac{d_{++} + d_{--}}{d_{++} + d_{+-} + d_{-+} + d_{--}}$$

$$J(x_i, x_j) = \frac{d_{++}}{d_{++} + d_{+-} + d_{-+}}$$

où d_{++} correspond au nombre de propriétés partagées par les deux objets, d_{--} au nombre de propriétés vérifiées par aucun des deux objets, etc. Ces deux indices[2] sont plutôt adaptés dans le cas d'attributs initialement binaires. Dans le cas contraire, la définition de nouvelles

[2]Il existe également d'autres indices utilisant ce type de comptage, comme par exemple les indices de Sorensen [173] ou de Sokal et Michener [172].

propriétés (binaires) impliquera une sur-représentation des attributs ayant de nombreuses modalités. L'indice *Rand* est alors le moins adapté pour cette dernière situation. Notons également que les attributs numériques sont généralement traités par discrétisation (découpage en intervalles).

Il découle de ce qui précède, que la redéfinition proposée de l'espace de description ne convient pas dans le cas d'attributs non-binaires et/ou numériques. D'autres approches ont alors été proposées pour un ensemble de données décrit par des attributs variés : numériques et symboliques, binaires ou non [118]. Nous choisissons de présenter une mesure récente, proposée par Martin et Moal [120] et basée sur la définition d'un nouveau langage de description $\mathcal{L}$ adapté. Étant donnés deux objets x_i et x_j, leur similarité est évaluée par :

$$s(x_i, x_j) = \frac{1}{|\mathcal{L}|} \sum_{t \in \mathcal{L}} \delta_t(x_i, x_j)$$

où t désigne un terme (propriété) du langage $\mathcal{L}$ et $\delta_t(x_i, x_j) = 0$ si l'un des deux objets seulement satisfait la propriété t et $\delta_t(x_i, x_j) = 1$ sinon. La particularité de cette mesure réside dans la définition des termes du langage ; plusieurs types de termes peuvent être envisagés :

- Pour les attributs numériques, des termes de la forme $att_i = val_j$, $att_i \leq val_j$, $att_i \geq val_j$ ou $val_k \leq att_i \leq val_j$, peuvent être envisagés, pour des valeurs val_j appartenant au domaine de définition de l'attribut considéré ou choisis parmi les valeurs observées dans l'ensemble des données,
- Pour des attributs symboliques, on définit généralement des termes de la forme $att_i = val_j$, où val_j correspond à l'une des modalités possibles pour l'attribut att_i.

Les termes du langage $\mathcal{L}$ peuvent être de nature plus complexe. On peut par exemple combiner plusieurs attributs sous la forme d'une expression linéaire (*e.g.* $a_0 + a_1 att_1 + \cdots + a_p att_p \geq 0$ pour des attributs tous numériques) ou d'une conjonction de termes simples (*e.g.* $att_i \leq val_j$ ET $att_k = val_l$). Le nouveau langage de description permet de traiter les attributs numériques sans discrétisation préalable, ainsi que les attributs symboliques en évitant une sur-représentation des attributs ayant de nombreuses modalités. Enfin on peut choisir d'intégrer des connaissances extérieures pour affiner la pertinence du langage, ces connaissances permettent de favoriser, par exemple, la mise en commun de modalités jugées "assez proches" (*e.g.* $couleur = rouge$ OU $couleur = rose$).

Cette fonction de similarité peut facilement donner lieu à une pseudo-distance (par transformation $d(x_i, x_j) = 1 - s(x_i, x_j)$), vérifiant les propriétés de minimalité, de symétrie et l'inégalité triangulaire, sans toutefois satisfaire la propriété d'identité puisque deux objets différents peuvent, selon le langage utilisé, satisfaire ou non les mêmes propriétés.

Cette section était destinée à présenter brièvement les différents choix de mesures de similarité, selon la nature des données à traiter. Nous choisirons dans la suite d'utiliser naturellement la distance euclidienne lorsque les données sont caractérisées uniquement par des variables numériques, et la mesure de Martin et Moal dans toute autre situation. Nous tâcherons cependant de justifier ces choix le moment venu, au regard des mesures traditionnellement utilisées dans les applications considérées.

1.5 Clustering "dur", clustering avec recouvrements et clustering "flou"

Le processus de clustering peut être formalisé de trois façons différentes, comme nous l'avons vu en section 1.2 : le clustering "dur", le clustering avec recouvrements et le clustering "flou". Chacun de ces trois formalismes correspond à un niveau de contraintes respectivement de moins en moins fortes.

Le clustering "dur" (*hard* ou *crisp-clustering* en anglais) est le plus utilisé. La grande majorité des travaux de synthèse sur le clustering n'évoque que cet aspect [88, 71, 10]. La contrainte sur les objets est très forte puisque chaque objet doit être classé dans une et une seule classe. Le résultat prend alors la forme de partitions ou hiérarchies strictes. L'avantage de ce formalisme est de proposer une organisation simple, sans ambiguïté et facilement exploitable pour d'autres traitements. Par exemple, si l'on souhaite proposer *a posteriori*, une définition en intension d'un groupe d'objets, ce type de schéma ("strict") est adapté.

Le clustering "flou" (*fuzzy-clustering* en anglais) correspond à une représentation plus souple de l'organisation des données. Dans ce formalisme, chaque objet x_i est affecté de façon "fractionnaire" c'est à dire qu'il se partage suivant les différents clusters $\{C_1, \ldots, C_t\}$ relativement à une famille de fonctions d'appartenance $\{u_j(x_i)\}_{j=1\ldots t}$. On parle aussi de fonctions d'affectation (*assignment* en anglais). Deux principaux modèles d'affectation existent [9] : les fonctions probabilistes ou possibilistes[3]. A la différence du modèle possibiliste, une fonction d'affectation probabiliste vérifie la contrainte suivante :

$$\forall x_i \in X, \sum_{j=1}^{t} u_j(x_i) = 1$$

Le clustering flou est un domaine de recherche à l'intersection de deux champs majeurs que sont le clustering et la théorie des ensembles flous. Ce type de formalisme permet de tenir compte des incertitudes et ambiguïtés pouvant intervenir dans l'appartenance d'un objet à une classe. L'utilisation d'algorithmes de clustering flou est très fréquente dans les domaines d'applications tels que le regroupement de données textuelles ou la segmentation d'images [22]. Dans le premier cas, les textes sont composés d'unités sémantiques (par exemple des mots) pouvant s'organiser autour de thèmes. Cependant, chaque mot peut se retrouver associé à différents thèmes, sans exclusivité et avec un degré d'implication plus ou moins fort. En segmentation d'images, différents niveaux de traitement conduisent à des données incertaines, qu'il s'agisse de la couleur d'un pixel, de sa luminosité, de la texture d'un bloc de pixel ou encore de sa forme géométrique.

Les clusters résultant d'un processus flou renferment donc une information plus riche que dans le cas du clustering dur. Cependant, le principal inconvénient de ce type d'approche concerne l'exploitation, la visualisation et l'interprétation des clusters, beaucoup plus difficiles.

Pour combiner les avantages des deux méthodes et en éviter les limitations, un troisième formalisme, intermédiaire, est proposé. Le clustering avec recouvrements propose une affectation dure de chaque objet à une ou plusieurs classes. Ces algorithmes sont parfois dénommés *soft-clustering* même si, assez souvent, la confusion est faite avec le clustering flou. Il existe très peu d'approches de clustering avec recouvrements. Notre étude se si-

[3] On parle aussi respectivement de fonctions relatives ou absolues.

tue dans ce cadre de recherches qui compte de récentes contributions applicables à des domaines précis [115, 137].

Dans la suite de ce chapitre, nous nous attacherons à étudier les trois formalismes précédents (clustering "dur", clustering avec recouvrements et clustering "flou"), dans diverses approches de clustering.

1.6 Les différentes méthodes de clustering

Dans cette partie, nous dressons un panorama des différentes méthodes de clustering rencontrées dans la littérature. Il est difficile de proposer une classification "logique" de ces méthodes ; les différentes études de synthèse proposent chacune leur propre organisation. [91] présentent successivement les méthodes de clustering hiérarchique, par partitionnement et enfin les approches probabilistes, tandis que [10] choisit de considérer principalement les approches hiérarchiques et de partitionnement, en incluant les méthodes probabilistes dans cette dernière catégorie. Enfin, [64] distingue les méthodes de clustering paramétriques et non paramétriques. Ces différentes présentations des méthodes de clustering sont dues, d'une part, au fait que les classes d'algorithmes se recouvrent (certaines méthodes s'appuyant, par exemple, sur des modèles probabilistes proposent un partitionnement) et d'autre part, diffèrent selon que l'on s'intéresse plutôt aux résultats du clustering (hiérarchie *vs.* partitionnement, clustering dur *vs.* clustering flou etc.), ou à la méthode utilisée pour parvenir à ce résultat (utilisation de fonctions probabilistes *versus* utilisation de graphes, etc.).

Nous choisissons ici une classification proche de celle proposée par [91], en présentant d'abord les méthodes hiérarchiques et de partitionnement, puis en nous intéressant successivement aux approches basées sur des fonctions probabilistes, des découpages en grilles, les densités et des descriptions conceptuelles. Pour chaque type d'approche nous nous appuyons sur un algorithme "typique", que nous présentons globalement, et étudions l'extension possible de ces travaux au clustering avec recouvrement (*soft-clustering* ou *overlapping-clustering*) et au clustering flou (*fuzzy-clustering*).

1.6.1 Le clustering hiérarchique

Le principe des algorithmes hiérarchiques est de construire un arbre de clusters (ou dendrogramme) tel que présenté en figure 1.3 et formalisé par la définition 1.4 :

- la racine de l'arbre est formée par le cluster X contenant l'ensemble des objets,
- chaque nœud de l'arbre constitue un cluster $C_i \subset X$,
- les feuilles de l'arbre correspondent aux singletons $\{x_1\}, \ldots, \{x_n\}$,
- l'union des objets contenus dans les fils d'un nœud donné, correspond aux objets présents dans ce nœud,
- les "paliers" sont indicés relativement à l'ordre de construction[4].

Ces hiérarchies sont généralement appréciées puisqu'elles permettent une visualisation de l'organisation des données et du processus de clustering. A partir de ce dendrogramme, il est possible d'obtenir une partition de X en coupant l'arbre à un niveau l donné. Par exemple, le choix de $l = 4$ dans le dendrogramme de la figure 1.3 renvoie le partitionnement suivant : $\mathcal{C} = \{\{x_1, x_2, x_3\}, \{x_4, x_5, x_6\}\}$. Ce dernier paramètre l peut être choisi

[4]On peut également choisir d'indicer les paliers d'une hiérarchie relativement à la similarité entre les deux sous-paliers.

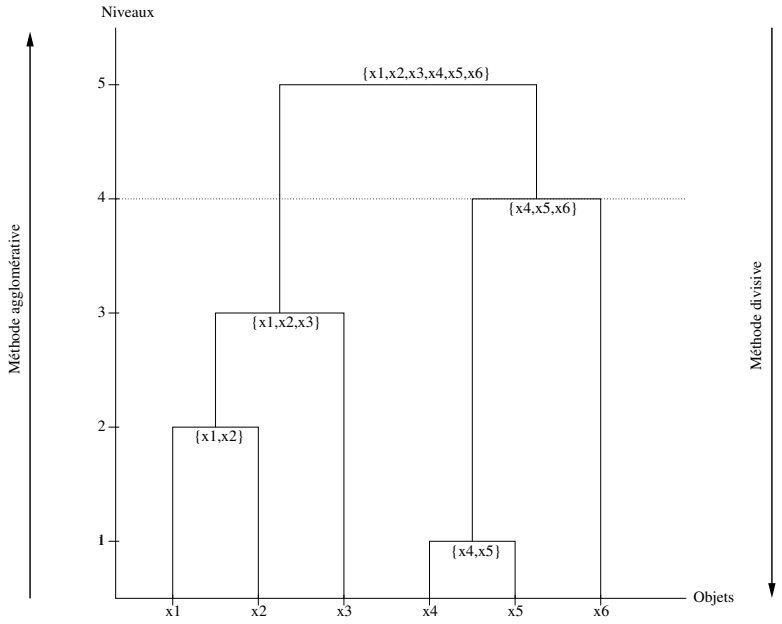


FIG. 1.3 – Exemple de dendrogramme.

relativement au nombre de clusters désiré ou à l'aide d'une analyse statistique de la qualité des différentes partitions que l'on peut extraire de l'arbre (cf. 1.7).

On distingue deux approches pour parvenir à un tel arbre hiérarchique : les algorithmes agglomératifs et divisifs. Un regroupement agglomératif[5] construit l'arbre en partant des "feuilles" (singletons) et procède par fusions successives des plus proches clusters jusqu'à obtenir un unique cluster "racine", contenant l'ensemble des objets. Par opposition, les algorithmes divisifs[6] considèrent d'abord la "racine" contenant tous les objets, puis procèdent par divisions successives de chaque nœud jusqu'à obtenir des singletons. Notons que pour chacune de ces deux méthodes, l'arbre hiérarchique n'est pas nécessairement construit totalement. Le processus peut être stoppé lorsque le nombre de clusters désiré est atteint ou lorsqu'un seuil de qualité est dépassé (*e.g.* distance maximale de fusion).

Les hiérarchies strictes

Nous présentons ici les principaux algorithmes de clustering par construction de hiérarchies strictes telles que définies en 1.4. Il existe très peu d'algorithmes divisifs, notamment à cause de la difficulté à définir un critère de séparation d'un cluster. En effet pour un cluster de taille n, il y a $2^{n-1}-1$ possibilités pour diviser ce cluster en deux sous-clusters. Dans le cas agglomératif, chaque fusion de 2 clusters parmi n, offre seulement $\frac{n(n-1)}{2}$ possibilités.

Pour éviter d'explorer toutes les possibilités de divisions, l'algorithme DIANA (*DIvisive ANAlysis*) [97], présenté en figure 1.4, recherche d'abord l'objet le plus "atypique" du cluster avant de lui agréger éventuellement d'autres objets proches de façon à distinguer deux sous-clusters. D'autres techniques ont été envisagées telles que la séparation par décomposition en k valeurs propres, utilisée dans l'algorithme PDDP (*Principal Direction*

[5] On parle aussi d'approche *bottom-up* puisque l'arbre se construit du "bas" vers le "haut".
[6] On parle aussi d'approche *top-down* puisque l'arbre se construit du "haut" vers le "bas".

Algorithme DIANA : *DIvisive ANAlysis*

Entrée : Une matrice de similarité S sur l'ensemble des objets à traiter X
Sortie : Une hiérarchie P

1. Initialisation à 1 cluster (racine), $\mathcal{C} = \{\{x_1, \ldots, x_n\}\}$, et $P = \mathcal{C}$
2. Sélectionner le cluster $C \in \mathcal{C}$ de diamètre maximum
3. Identifier dans C l'objet (ou l'un des objets) x^* ayant la plus faible similarité moyenne avec les autres objets :

$$x^* = \arg\min_{x_i \in C} \frac{1}{|C| - 1} \sum_{j \neq i} s(x_i, x_j)$$

x^* initialise un nouveau cluster C^*,
4. Pour chaque objet $x_i \notin C^*$ calculer :

$$s_i = [\textit{moyenne des } s(x_i, x_j),\ x_j \in C \setminus C^*] - [\textit{moyenne des } s(x_i, x_j),\ x_j \in C^*]$$

5. Soit x_k l'objet pour lequel s_k est minimal. Si s_k est négatif alors ajouter x_k à C^*
6. Répéter les étapes 3 et 4 jusqu'à $s_k > 0$
7. Remplacer C par $C \backslash C^*$ et C^* dans $\mathcal{C}$ puis ajouter $C \backslash C^*$ et C^* dans P
8. Répéter les étapes 2 à 7 jusqu'à ce que chaque cluster de $\mathcal{C}$ soit réduit à un singleton
9. Retourner P, un ensemble de parties non vides sur X, correspondant aux nœuds de la hiérarchie

FIG. 1.4 – Algorithme divisif hiérarchique DIANA.

Divisive Partitioning) [16] ou le recours à une méthode de partitionnement du type k-moyennes (cf. 1.6.2) à chaque nœud de la hiérarchie.

En revanche, les algorithmes agglomératifs hiérarchiques sont nombreux. Les premières études ont proposé les algorithmes SLINK (*Single-LINK*), CLINK (*Complete-LINK*) et ALINK (*Average-LINK*), tous trois basés sur le procédé SAHN (*Sequential Agglomerative Hierarchical and Non-overlapping* [7]) [171], présenté en figure 1.5. La méthode utilisée pour évaluer la similarité entre deux clusters différencie ces trois algorithmes :

- SLINK : la similarité entre deux clusters C_a et C_b est déterminée par la similarité entre les deux objets $(x_i, x_j) \in C_a \times C_b$ les plus similaires,
- CLINK : la similarité entre deux clusters C_a et C_b est déterminée par la similarité entre les deux objets $(x_i, x_j) \in C_a \times C_b$ les moins similaires,
- ALINK : la similarité entre deux clusters C_a et C_b est déterminée par la moyenne des similarités entre deux objets $(x_i, x_j) \in C_a \times C_b$.

Algorithme SAHN : *Sequential Agglomerative Hierarchical and Non-overlapping*

Entrée : Une matrice de similarité S sur l'ensemble des objets à traiter X
Sortie : Une hiérarchie P

1. Initialisation à n singletons, $\mathcal{C} = \{\{x_1\}, \ldots, \{x_n\}\}$, et $P = \mathcal{C}$
2. Identifier dans $\mathcal{C}$ les deux objets C_k et C_l les plus proches selon S :

$$(C_k, C_l) = \underset{(C_i, C_j) \in\ \mathcal{C}^2}{\arg\max} \{sim(C_i, C_j)\}$$

3. Remplacer $\{C_k\}$ et $\{C_l\}$ par $\{C_k \cup C_l\}$ dans $\mathcal{C}$ et ajouter $\{C_k \cup C_l\}$ à P
4. Recalculer la matrice S en conséquence
5. Si $\mathcal{C} \neq \{\{x_1, \ldots, x_n\}\}$ revenir à l'étape 2.
6. Retourner P, un ensemble de parties non vides sur X, correspondant aux nœuds de la hiérarchie

FIG. 1.5 – Algorithme agglomératif hiérarchique SAHN.

Des travaux plus récents ont permis de proposer des algorithmes agglomératifs hiérarchiques permettant d'obtenir des clusters de formes variées (contrairement aux algorithmes CLINK et ALINK), tels que l'algorithme CURE (*Clustering Using Representatives*) [74] et surtout l'algorithme CHAMELEON [95].

La principale caractéristique de l'algorithme CURE est de proposer une méthode originale d'indexation des clusters. Plutôt que d'indexer un cluster par l'ensemble des objets qui le composent (indexation graphique) ou par un centroïde (indexation géométrique), CURE détermine un nombre constant d'objets représentatifs de ce cluster.

CHAMELEON procède en deux étapes : une première étape consiste à partitionner l'ensemble des objets en petits clusters homogènes, puis une phase agglomérative per-

[7] SAHN est également parfois dénommé *Sequential Agglomerative Hierarchical and Nested*

met d'aboutir à une hiérarchie. Pour cela, CHAMELEON s'appuie sur un formalisme de représentation basé sur les graphes (cf. 1.6.7).

Enfin des méthodes telles que ROCK (*RObust Clustering using linKs*) [75] et COBWEB [60] construisent un arbre à partir de données décrites par des variables qualitatives.

Les pseudo-hiérarchies

Les "pyramides", introduites par E. Diday [46], constituent une généralisation des approches hiérarchiques précédentes. La motivation première est la construction de "classes recouvrantes" avec la contrainte de pouvoir visualiser l'organisation des données. Les pyramides constituent alors ce que l'on appelle aussi des pseudo-hiérarchies au sens où nous l'avons défini en 1.5. Une pyramide telle que nous la présentons en figure 1.6 autorise chaque nœud à avoir jusqu'à deux prédécesseurs. Lorsque l'on coupe la pyramide à un niveau l donné, les clusters obtenus forment alors une pseudo-partition de X. Dans la figure 1.6, en choisissant de couper le dendrogramme au niveau matérialisé par les pointillés, la pseudo-partition $\mathcal{C} = \{\{x_1, x_2, x_3, x_4\}, \{x_3, x_4, x_5\}, \{x_6\}\}$ est obtenue.

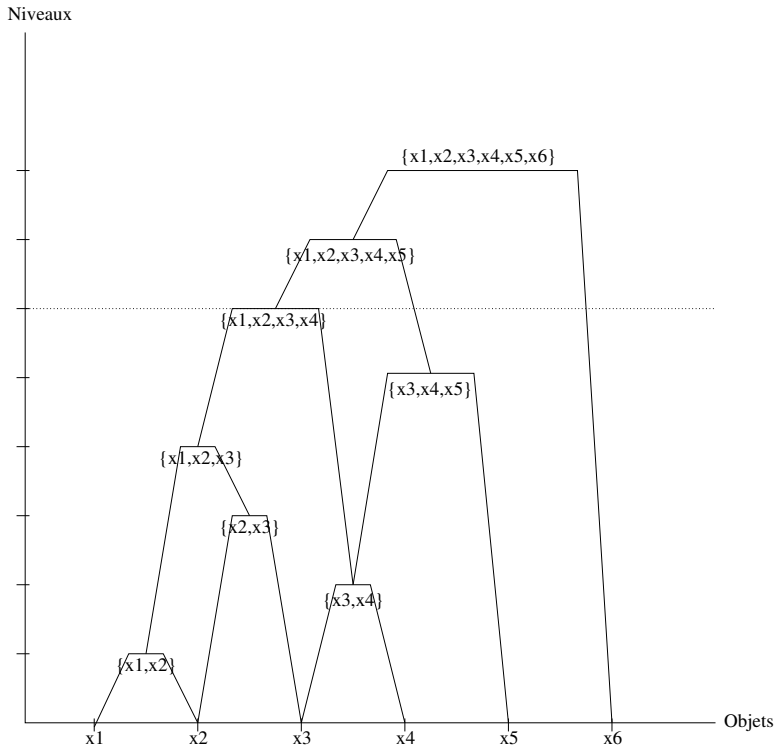


FIG. 1.6 – Exemple de pyramide.

L'algorithme CAP (*Classification Ascendante Pyramidale*) a été proposé pour construire une pyramide ; nous présentons le principe global de cet algorithme dans la figure 1.7.

Dans l'algorithme CAP, la condition *i*) assure que chaque nœud de la pyramide aura au plus deux prédécesseurs. Les conditions *ii*) et *iii*) permettent d'éviter les croisements illustrés en figure 1.8.

Dans le premier exemple proposé en figure 1.8 (pyramide de gauche), la condition *ii*) n'est pas vérifiée puisque C_1 et C_2 sont agrégés alors qu'ils ne sont pas connexes à cause

Algorithme CAP : *Classification Ascendante Pyramidale*

Entrées : Une matrice de similarité S sur l'ensemble des objets à traiter X ainsi que $\leq_X$, un ordre sur X
Sortie : Une pyramide P

1. Initialisation à n singletons, $P = \{\{x_1\}, \ldots, \{x_n\}\}$
2. On agrège les deux groupes $C_i,\ C_j \in P$ les plus proches selon S, et vérifiant les conditions suivantes :
   i) C_i et C_j ont été agrégés au plus une fois,
   ii) $C_i \cup C_j$ est connexe selon $\leq_X$,
   iii) Soit $C_k \in P$ tel que $C_i \subset C_k$ alors C_i contient l'une des extrémités de la partie connexe C_k
3. Ajouter $C_i \cup C_j$ dans P
4. On recommence l'étape 2 jusqu'à ce qu'un groupe de P soit égal à X
5. Retourner P, un ensemble de parties non vides sur X, correspondant aux nœuds de la pyramide

FIG. 1.7 – Algorithme de construction d'une pyramide.

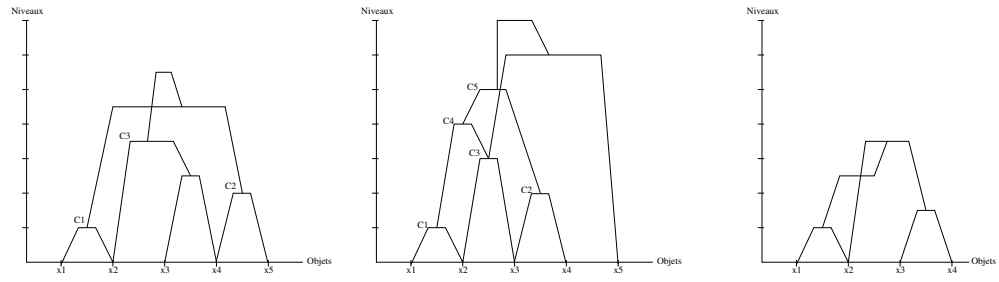


FIG. 1.8 – Exemples de croisements (gauche et centre) et d'inversions (droite) dans les pyramides.

de C_3. De même, dans le second exemple (pyramide du centre), c'est la condition *iii*) qui cette fois n'est pas vérifiée puisque $\{x_5\}$ est agrégé à C_3 alors que $C_3 \subset C_5$ et C_3 ne contient ni x_1, ni x_4, extrémités de C_5.

Par la suite, plusieurs extensions de l'algorithme CAP ont été proposées. Notons, par exemple, l'algorithme PYRAM [68] qui permet de générer des pyramides plus simples, à savoir sans inversions (cf. figure 1.8), ou encore les algorithmes CAPS et CAPSO [157] qui permettent de générer des pyramides symboliques, où chaque nœud est défini en intension. Enfin, certains travaux actuels concernent l'extension des pyramides présentées jusqu'ici aux pyramides 3D et plus.

Les hiérarchies floues

Les hiérarchies floues sont généraelemnt peu utilisées. On note cependant quelques applications dans le domaine de la segmentation d'images ainsi que dans les formalismes

de représentation notamment sémantique. Nous présentons deux types de hiérarchies floues chacune liée à l'application ou à l'algorithme de construction.

Algorithme HFCM : *Hierarchical Fuzzy-k-Means*

Entrées : L'ensemble des objets à traiter X, une mesure de qualité ϕ ainsi qu'un seuil de qualité α
Sortie : Une hiérarchie floue P

1. Initialisation à un cluster (racine) $\mathcal{C} = \{X\}$ et $P = \mathcal{C}$
2. Pour chaque cluster $C_i \in \mathcal{C}$ tel que $\phi(C_i) < \alpha$:
    3. Appliquer l'algorithme FCM en recherchant le nombre optimal k_i de clusters
$$C_i \rightarrow \{C_{i,1}, \ldots, C_{i,k_i}\}$$
    4. Supprimer C_i dans $\mathcal{C}$ et ajouter $C_{i,1}, \ldots, C_{i,k_i}$ dans $\mathcal{C}$ et dans P
5. Recommencer l'étape 2 jusqu'à ce que tous les clusters C_i dans $\mathcal{C}$ soient tels que $\phi(C_i) > \alpha$
6. Retourner P, un ensemble de parties floues non vides sur X, correspondant aux nœuds de la hiérarchie

FIG. 1.9 – Algorithme divisif des k-moyennes flou hiérarchique.

Dans le cas de la segmentation d'images, on a généralement recours à un algorithme du type k-moyennes flou hiérarchique [54]. L'algorithme des k-moyennes est une méthode de partitionnement que nous présenterons dans la section suivante. L'algorithme des k-moyennes flou FCM (*fuzzy-k-means*) en est une extension. Enfin, la méthode des k-moyennes flou hiérarchique consiste à exécuter l'algorithme FCM itérativement sur chaque nœud de la hiérarchie, en commençant par la racine constituée de tous les éléments, et jusqu'à obtenir des clusters flous valides relativement à une mesure de qualité. Le nombre k de groupes étant le principal paramètre en entrée de l'algorithme FCM, il peut être proposé de rechercher le nombre optimal de clusters k à l'aide d'un indice, tel que l'indice de *Dunn* ou de *Davies-Bouldin*, tous deux définis dans l'annexe A. Nous présentons en figure 1.9 l'algorithme des k-moyennes flou hiérarchique, que nous notons HFCM (*Hierarchical Fuzzy-k-Means*).

Il s'agit alors d'une méthode hiérarchique divisive. Cependant, contrairement aux hiérarchies strictes et aux pseudo-hiérarchies, les feuilles du dendrogramme ne correspondent pas aux singletons $\{\{x_1\}, \ldots, \{x_n\}\}$ mais à des clusters flous validés par une mesure de qualité. De même, par l'algorithme HFCM, le dendrogramme obtenu n'est pas un arbre binaire puisque chaque nœud peut avoir plus de deux successeurs. La figure 1.10 présente un exemple de dendrogramme-flou obtenu par l'algorithme HFCM. Dans ce dendrogramme, chaque nœud de l'arbre est constitué par l'ensemble des objets de X avec un certain degré d'appartenance (valeurs précisées entre parenthèses). Notons que pour un objet x_1 donné, sa valeur d'appartenance $u_2(x_1)$ à un nœud C_2 de l'arbre est égale à la somme de ses valeurs d'appartenance aux successeurs directes de ce nœud ($u_2(x_1) = u_3(x_1) + u_4(x_1) + u_5(x_1)$).

L'objectif premier du clustering hiérarchique est de permettre la visualisation du pro-

cessus, cependant il apparaît clairement que la représentation des dendrogrammes flous est difficile puisque chaque nœud de la hiérarchie doit comporter un vecteur de taille n. Finalement, ce dendrogramme aide peu à la compréhension du processus de clustering.

Une autre forme de hiérarchie floue intervient, par exemple, dans le cadre de la représentation de concepts sémantiques sous forme d'une hiérarchie (thésaurus). Ce formalisme de représentation peut également être utilisé en fouille de données dans des applications visant à synthétiser l'information contenue dans une base de données [107]. On parle communément de hiérarchies "IS A", illustrées par la figure 1.11.

Généralement construite de manière agglomérative, la hiérarchie "IS A" est un dendrogramme flou tel que les feuilles sont définies par les singletons $\{x_1\}, \ldots, \{x_n\}$. Chaque nœud de l'arbre (y compris la racine) correspond à un objet représentatif (centroïde, médoïde, etc.) du cluster flou, défini par l'ensemble de ses successeurs, et muni d'un degré d'appartenance[8]. Une nouvelle fois, la faible utilisation de ce type de classification en apprentissage non-supervisé s'explique par la difficulté à interpréter le dendrogramme obtenu.

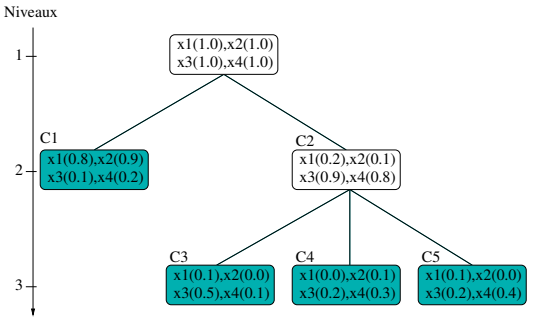


Fig. 1.10 – Exemple de dendrogramme flou.

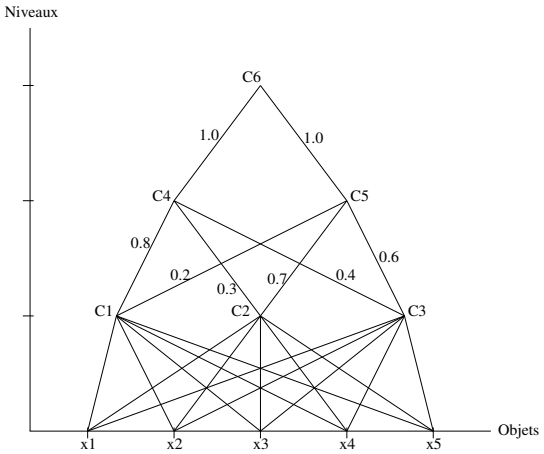


Fig. 1.11 – Exemple de Hiérarchie "IS A".

1.6.2 Le clustering par partitionnement

Contrairement aux approches hiérarchiques précédentes, les algorithmes de partitionnement proposent, en sortie, une partition de l'espace des objets plutôt qu'une structure organisationnelle du type "dendrogramme". Le principe est alors de comparer plusieurs schémas de clustering (plusieurs partitionnements) afin de retenir le schéma qui optimise un critère de qualité. En pratique il est impossible de générer tous les schémas de clustering pour des raisons évidentes de complexité. On cherche alors un "bon" schéma correspondant à un optimum (le plus souvent "local") pour ce critère. Cet optimum est obtenu de façon itérative, en améliorant un schéma initial choisi plus ou moins aléatoirement, par ré-allocation des objets autour de centres mobiles. Nous étudions, dans cette section, les différentes techniques de ré-allocation à partir de l'algorithme bien connu des k-moyennes et dans un souci constant d'évoluer d'une partition stricte vers une pseudo-partition, puis vers une partition floue.

[8]Pour des raisons de clarté, nous n'avons pas reporté dans la figure 1.11, les valeurs d'appartenance sur les branches du premier niveau du dendrogramme.

Le partitionnement strict

L'algorithme des k-moyennes (k-*means*) [117] est sans aucun doute la méthode de partitionnement la plus connue et la plus utilisée dans divers domaines d'application. Ce succès est dû au fait que cet algorithme présente un rapport coût/efficacité avantageux. Nous présentons l'algorithme des k-moyennes en figure 1.12.

Algorithme k-moyennes :

Entrées : k le nombre de clusters désiré, d une mesure de dissimilarité sur l'ensemble des objets à traiter X
Sortie : Une partition $\mathcal{C} = \{C_1, \ldots, C_k\}$

Étape 0 : 1. Initialisation par tirage aléatoire dans X, de k centres $x^*_{1,0}, \ldots, x^*_{k,0}$
2. Constitution d'une partition initiale $\mathcal{C}_0 = \{C_1, \ldots, C_k\}$ par allocation de
chaque objet $x_i \in X$ au centre le plus proche :

$$C_l = \{x_i \in X | d(x_i, x^*_{l,0}) = \min_{h=1,\ldots,k} d(x_i, x^*_{h,0})\}$$

3. Calcul des centroïdes des k classes obtenues $x^*_{1,1}, \ldots, x^*_{k,1}$
(cf. définition 1.8)

Étape t : 4. Constitution d'une nouvelle partition $\mathcal{C}_t = \{C_1, \ldots, C_k\}$ par allocation de chaque objet $x_i \in X$ au centre le plus proche :

$$C_l = \{x_i \in X | d(x_i, x^*_{l,t}) = \min_{h=1,\ldots,k} d(x_i, x^*_{h,t})\}$$

5. Calcul des centroïdes des k classes obtenues $x^*_{1,t+1}, \ldots, x^*_{k,t+1}$

6. Répéter les étapes 4 et 5 tant que des changements s'opèrent d'un schéma $\mathcal{C}_t$ à un schéma $\mathcal{C}_{t+1}$ ou jusqu'à un nombre τ d'itérations
7. Retourner la partition finale $\mathcal{C}_{finale}$

FIG. 1.12 – Algorithme de partitionnement k-moyennes.

A partir d'un tirage aléatoire de k "graines" dans X, l'algorithme des k-moyennes procède par itérations de super-étapes d'allocations des objets $\{x_1, \ldots, x_n\}$ aux centres (initialement les graines), suivies du calcul de la position des nouveaux centres, dits "mobiles". On peut montrer que, par ce processus, le critère de *variance intra-cluster*[9] converge vers un minimum local, dépendant de l'initialisation de l'algorithme. Rappelons que ce critère est défini par

$$V(t) = \sum_{h=1}^{k} \sum_{i=1}^{n} d(x_i, x^*_{h,t})^2$$

où $x^*_{h,t}$ désigne le centroïde du cluster C_h à l'étape t.

[9] On parle qussi du critère des moindres carrés.

Une autre version de l'algorithme des k-moyennes consiste à recalculer les centres mobiles au fur et à mesure des ré-allocations. Ainsi, chaque ré-allocation d'un objet à un centre mobile peut engendrer une modification immédiate sur les deux centres concernés : le centre auquel l'objet est nouvellement alloué et le centre auquel l'objet était anciennement alloué. Cette version, dite "adaptative", permet, en pratique, de parvenir plus rapidement à une partition de bonne qualité. Cependant le résultat dépend cette fois de l'ordre de traitement des objets.

L'algorithme des k-moyennes ne peut être utilisé que sur des données décrites par des attributs numériques permettant ainsi le calcul des centroïdes. Dans le cas d'attributs symboliques (ou catégoriels) plusieurs variantes ont été proposées : l'algorithme des k-médoïdes, l'algorithme PAM (*Partitioning Around Medoids*) [96] ou encore la méthode des nuées dynamiques [45]. L'algorithme des k-médoïdes se différencie de la méthode des k-moyennes par l'utilisation de médoïdes plutôt que de centroïdes pour représenter les classes. Enfin, la méthode des nuées dynamiques représente chaque classe par un "noyau", constitué d'un certain nombre d'objets de cette classe, souvent plus représentatif qu'un objet unique.

La recherche des médoïdes étant plus coûteuse que le simple calcul de centroïdes, des extensions de l'algorithme k-médoïdes ont été proposées dans le cas du traitement de larges ensembles de données. Notons par exemple la méthode CLARA (*Clustering LARge Applications*) [97] qui procède par échantillonnage de l'ensemble des données en itérant plusieurs fois les étapes suivantes :

- tirage aléatoire de $40 + 2k$ objets[10],
- exécution de l'algorithme PAM sur l'échantillon obtenu.

Finalement, la partition finale est obtenue par affectation de tous les objets aux médoïdes issus du meilleur schéma généré.

Enfin, l'algorithme CLARANS (*CLustering Algorithm based on RANdomized Search*) [136] propose une méthode originale de recherche d'un ensemble optimal de k médoïdes. A partir d'un graphe, où chaque nœud correspond à un schéma de clustering différent (ensemble de k médoïdes), l'algorithme commence par choisir un nœud au hasard puis parcourt le graphe de proche en proche jusqu'à observer un minimum local. Dans ce graphe, deux nœuds sont voisins s'ils ne diffèrent que d'un médoïde. Ce processus est itéré plusieurs fois, et le meilleur schéma est retourné.

Nous remarquons que toutes les méthodes précédentes nécessitent la connaissance du paramètre k, correspondant au nombre de groupes attendus en sortie du processus de clustering. La méthode ISODATA (*Iterative Self-Organizing Data Analysis Technique*) [8] est parfois utilisée pour affiner les clusters obtenus par un algorithme de partitionnement. Cette technique consiste à éclater ou fusionner les clusters en fonction de seuils préétablis. Par exemple, on fusionnera deux groupes C_i et C_j si leur distance inter-centre $d(x_i^*, x_j^*)$ est inférieure à un certain seuil α. De même, on éclatera un cluster C_i d'inertie supérieure à un seuil donné β. Dans cette famille d'algorithmes, le nombre de clusters attendus n'est pas fixé *a priori*.

Pseudo-partitions et partitions floues

Parmi les méthodes de clustering procédant par ré-allocations, il n'existe pas d'extension permettant directement la construction de pseudo-partitions. Pour obtenir une

[10] Les expérimentations présentées dans [97] indiquent que cinq échantillons de $40 + 2k$ objets permet d'obtenir des résultats satisfaisants.

pseudo-partition, on considère généralement un processus de clustering flou, le schéma obtenu étant ensuite restreint par le biais de seuils d'affectation.

Nous choisissons de présenter en figure 1.13 la variante floue de la méthode des k-moyennes, à savoir l'algorithme des k-moyennes flou (*Fuzzy k-means*), proposé initialement par [50], puis généralisé par [11].

Algorithme k-moyennes flou :

Entrées : k le nombre de clusters désiré, d une mesure de dissimilarité sur l'ensemble X des objets à traiter, T le nombre maximum d'itérations, un poids $m > 1$ et un seuil $\epsilon > 0$
Sortie : Une partition floue $\mathcal{C} = \{C_1, \ldots, C_k\}$ définie par les fonctions d'appartenance $\{u_h\}_{h=1\ldots k}$

1. (t=0) Choisir ou tirer aléatoirement une partition initiale $\{u_{h,t}\}_{h=1\ldots k}$
2. Calculer les centres de gravité $x^*_{1,t}, \ldots, x^*_{k,t}$ de chacune des k classes (à l'instant t)

$$v_j(x^*_{h,t}) = \frac{1}{\sum\limits_{x_i \in X} [u_{h,t}(x_i)]^m} . \sum_{x_i \in X} [u_{h,t}(x_i)]^m . v_j(x_i)$$

3. Calculer les nouvelles valeurs d'appartenance $\{u_{h,t+1}(x_i)\}_{h=1\ldots k}$ de chaque objet x_i à chaque centre de classe $x^*_{h,t}$

$$u_{h,t+1}(x_i) = \frac{[d(x_i, x^*_{h,t})]^{2/(1-m)}}{\sum\limits_{h=1}^{k} [d(x_i, x^*_{h,t})]^{2/(1-m)}}$$

4. Calculer les centres de gravité de chaque classe $x^*_{1,t+1}, \ldots, x^*_{k,t+1}$ de chacune des k classes (à l'instant $t+1$)
5. Calculer le déplacement global $E_t = \sum_{h=1}^{k} d(x^*_{h,t+1}, v_{h,t})$
6. Si $E_t \leq \epsilon$ alors retourner la partition floue définie par $\left(\{x^*_{h,t+1}, u_{h,t+1}\}_{h=1,\ldots,k}\right)$), sinon ($t$=$t$+1) retourner en 3)

FIG. 1.13 – Algorithme de partitionnement k-moyennes flou.

L'algorithme des k-moyennes flou est alors fondé sur le principe d'optimisation itérative d'un critère de variabilité intra-classe :

$$J_m = \sum_{i=1}^{n} \sum_{h=1}^{k} [u_h(x_i)]^m . d(x_i, x^*_h)$$

Le paramètre m est un paramètre appelé *fuzzifier* permettant d'amplifier les différences entre les objets éloignés et les objets centraux d'une même classe. Ce paramètre peut prendre sa valeur dans $]1, \infty[$. Quand $m \to 1$ les fonctions d'appartenance prennent leurs valeurs dans l'ensemble binaire $\{0, 1\}$, conduisant ainsi à une partition stricte des données. En revanche, lorsque $m \to \infty$ les valeurs d'appartenance convergent vers une distribution

uniforme $\{u_j(x_i)\}_{j=1...k} \rightarrow \frac{1}{k}$ et la partition obtenue est alors trop floue pour y rechercher une quelconque organisation des données. Les paramètres $m = 2$ ou $m = 1.25$ sont généralement choisis.

Algorithme k-médoïdes flou :

Entrées : k le nombre de clusters désiré, d une mesure de dissimilarité sur X l'ensemble des objets à traiter, T un nombre maximum d'itérations, un poids $m > 1$ et un seuil $\epsilon > 0$
Sortie : Une partition floue $\mathcal{C} = \{C_1, \ldots, C_k\}$ définie par les médoïdes des classes $x_1^*, \ldots, x_k^*$

1. (t=0) Choisir ou tirer aléatoirement k médoïdes $x_{1,t}^*, \ldots x_{k,t}^*$ dans X
2. Calculer les valeurs d'appartenance $\{u_{h,t}(x_i)\}_{h=1...k}$ de chaque objet x_i à chaque médoïde $x_{h,t}^*$

$$u_{h,t}(x_i) = \frac{(d(x_i, x_{h,t}^*)^{-1/(m-1)})}{\sum_{l=1}^{k}(d(x_i, x_{l,t}^*))^{-1/(m-1)}}$$

3. Recalculer les médoïdes $x_{1,t+1}^*, \ldots, x_{k,t+1}^*$:

$$x_{h,t+1}^* = \arg\min_{x_i \in X} \sum_{l=1}^{n} u_{h,t}(x_l)^m . d(x_i, x_l)$$

4. Si $\forall h,\ x_{h,t+1}^* = x_{h,t}^*$ ou si $t \geq T$ alors retourner la partition floue définie par $\left(\{x_{h,t+1}^*, u_{h,t+1}\}_{h=1,\ldots,k}\right)$) sinon $(t = t + 1)$ retourner en 2)

FIG. 1.14 – Algorithme de partitionnement k-médoïdes flou.

Plusieurs extensions de cet algorithme ont été proposées afin de considérer des données relationnelles[11] et d'optimiser un critère (ou fonction objective) plus robuste. En ce sens, l'algorithme des k-moyennes flou relationnel (*Relational Fuzzy k-Means*) [79] consiste à minimiser itérativement le critère suivant :

$$J_m = \sum_{l=1}^{k} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} u_l(x_i)^m u_l(x_j)^m . d(x_i, x_j)}{2. \sum_{j=1}^{n} u_l(x_j)^m}$$

Ce critère correspond à la somme sur chaque classe, de la *distance intra-classe moyenne*. Dans le cas $m = 2$, le modèle flou se rapproche de l'algorithme FANNY (*Fuzzy ANalYsis*)

[11] Rappelons que le terme "données relationnelles" est utilisé ici pour spécifier un ensemble de données où chaque objet est uniquement caractérisé par ses relations de (dis)similarités avec les autres objets (contrairement aux "données objets", caractérisées sur un ensemble de variables descriptives).

[97]. Cependant, ces derniers algorithmes restent coûteux puisque leur complexité est quadratique sur le nombre n d'objets dans X.

Nous présentons finalement (figure 1.14) l'algorithme des k-médoïdes flou (*Fuzzy c-Medoids*) [102] qui peut être considéré comme la version floue de l'algorithme des k-médoïdes. La complexité de cet algorithme est en $O(n^2)$ au pire cas, sachant qu'en pratique, le coût est moins élevé.

Une dernière méthode de partitionnement flou, inspirée de l'algorithme des k-moyennes, a été proposée par A. Lelu [110] sous la dénomination de k-moyennes axiales (*Axial k-means*). Cet algorithme consiste à représenter chaque objet de X par un vecteur normalisé. Tous les objets et centres de classes se trouvent alors sur une hyper-sphère de rayon 1. Au final, chaque classe est représentée au moyen d'un seul axe, et le degré d'appartenance d'un objet à une classe est obtenu par projection de l'objet sur l'axe correspondant.

La méthode des k-moyennes axiales est généralement employée dans le cadre du regroupement de termes afin de constituer des thèmes. La construction de classes recouvrantes (donc d'une pseudo-partition) s'effectue par la donnée d'un seuil, dit de *typicité*, tel que chaque objet x_i est affecté à une classe C_j si la projection $P_j(x_i)$ de x_i sur l'axe représentatif de la classe C_j dépasse ce seuil.

Comme nous l'avons précisé en début de section, c'est par le recours à un seuil, qu'une pseudo-partition peut être dérivée d'un schéma flou. Notons que [98] discutent de méthodes d'affectations strictes ou non (*hard-assignment* vs. *soft-assignment*). Comme dans la plupart des cas, la dénomination "soft-assignment" correspond ici à une affectation fractionnaire plus proche de ce que nous avons définie comme une affectation floue.

1.6.3 Le clustering par mélange de densités de probabilités

L'idée de base de ces méthodes est de considérer que les observations (ou objets) $x_1, \ldots, x_n$ ont été générées par un ensemble (ou mélange) de k distributions de probabilités. Le nombre k correspondant au nombre de clusters, il peut être connu *a priori* ou à définir en comparant les schémas issus de plusieurs valeurs de k. Des hypothèses sont faites sur la nature de ces distributions : l'hypothèse la plus courante est de considérer que les observations sont issues d'une loi normale, les densités de probabilités sont donc des gaussiennes. Il s'agit alors d'estimer les paramètres permettant de définir chaque distribution afin de coller au mieux avec les observations. Pour estimer ces paramètres, Rao [153] est le premier à utiliser la méthode du *maximum de vraisemblance* dans le cas de mélanges gaussiens.

Nous présentons en figure 1.15 l'algorithme EM *Expectation-Maximization* [40] simplifié, qui reprend l'approche générale d'estimation de gaussiennes par le maximum de vraisemblance ($\mathcal{L}$). Dans l'algorithme, l'hypothèse de distributions gaussiennes suppose l'estimation des paramètres suivants :

- $\{\mu_h\}_{h=1\ldots k}$: Les moyennes des k gaussiennes,
- $\{\sigma^2{}_h\}_{h=1\ldots k}$: Les variances des k gaussiennes.

Un dernier paramètre τ_h, indépendant du type de distribution choisie, correspond au paramètre de mélange.

Dans l'algorithme EM, la fonction à optimiser (ici à minimiser) correspond au log de vraisemblance négatif :

$$E = -\ln \mathcal{L} = -\sum_{i=1}^{n} \ln p(x_i) = -\sum_{i=1}^{n} \ln \sum_{h=1}^{k} \tau_h p(x_i|h)$$

Algorithme EM : *Expectation Maximization*

Entrées : L'ensemble des observations $X = \{x_1, \ldots, x_n\}$, le nombre k de clusters désirés et ϵ un seuil de tolérance
Sortie : $\{(\mu_h, {\sigma^2}_h, \tau_h)\}_{h=1\ldots k}$ un ensemble de k vecteurs de paramètres

1. Initialiser k vecteurs de paramètres $\{(\mu_h^0, {\sigma^2}_h^0, \tau_h^0)\}_{h=1\ldots k}$

2. A l'étape t, calculer les vecteurs $\{(\mu_h^{t+1}, {\sigma^2}_h^{t+1}, \tau_h^{t+1})\}_{h=1\ldots k}$ à partir des estimations de l'étape précédente ($\{(\mu_h^t, {\sigma^2}_h^t, \tau_h^t)\}_{h=1\ldots k}$) :

$$v_j(\mu_h^{t+1}) = \frac{\sum_{i=1}^n p^t(h|x_i).v_j(x_i)}{\sum_{i=1}^n p^t(h|x_i)}$$

$${\sigma^2}_h^{t+1} = \frac{\sum_{i=1}^n p^t(h|x_i)\|x_i - \mu_h^t\|^2}{\sum_{i=1}^n p^t(h|x_i)}$$

$$\tau_h^{t+1} = \frac{1}{n}\sum_{i=1}^n p^t(h|x_i)$$

3. Calculer la variation de la fonction d'erreur (log de vraisemblance négatif) :

$$\Delta^{t+1} = -\sum_{i=1}^n \ln(\frac{p^{t+1}(x_i)}{p^t(x_i)})$$

4. Si $\Delta > \epsilon$ alors retourner à l'étape 2, sinon retourner les vecteurs $\{(\mu_h^{t+1}, {\sigma^2}_h^{t+1}, \tau_h^{t+1})\}_{h=1\ldots k}$

N.B. $p^t(h|x_i) = \frac{p^t(x_i|h).\tau_h^t}{p^t(x_i)}$ par l'égalité de Bayes.

FIG. 1.15 – L'algorithme EM.

Notons que l'initialisation des paramètres (étape 1) influencera le résultat final de l'estimation. En effet, sachant qu'il est difficile d'aboutir à l'optimum global de la fonction de coût (l'espace des paramètres étant infini), l'algorithme EM permet d'obtenir un optimum local, lié à l'initialisation des paramètres.

L'algorithme EM est devenu quasiment incontournable, il est à l'origine de nombreux logiciels tels que Mclust-EMclust [61], EMmix [140] ou encore MIX-MOD [12]. Cependant, le principal inconvénient de cette méthode réside dans sa lenteur, due au nombre souvent élevé d'itérations nécessaires pour la convergence. Cet algorithme n'est donc pas approprié pour le traitement de grands volumes de données. Quelques travaux actuels s'attellent à accélérer les itérations, notons par exemple l'étude récente proposée dans [135].

L'algorithme AutoClass [21] adopte le principe de l'algorithme EM en proposant deux avancées majeures :

l'estimation du nombre k de clusters : le nombre de clusters est estimé par une analyse statistique bayésienne. Plusieurs modèles (avec différentes valeurs de k) sont

comparés relativement à un score bayésien qui mesure la probabilité, *a posteriori*, du modèle connaissant les données observées (supposées générées par ce modèle) ;

la prise en compte des données qualitatives : chaque type d'attribut donne lieu à un modèle statistique particulier :
- Les attributs quantitatifs réels sont modélisés par des distributions gaussiennes ou Log-gaussiennes,
- Les attributs qualitatifs sont modélisés par des distributions de Bernouilli,
- Les attributs quantitatifs à valeurs entières sont modélisés par des distributions de Poisson.

Globalement les approches de clustering par mélange de densités sont adaptées au traitement des données manquantes ou cachées. En effet, ces algorithmes se fondent sur l'hypothèse selon laquelle les données ont été générées par certaines lois de probabilités, dont seulement quelques observations sont fournies. Ces algorithmes supposent donc l'existence de données manquantes (non observées).

Cependant, ces méthodes nécessitent de formuler des hypothèses sur la distribution des observations (distributions gaussiennes par exemple). De telles hypothèses sont rarement vérifiées dans les applications réelles.

Enfin, les algorithmes EM et AutoClass précédemment cités, illustrent un autre inconvénient inhérent à ces méthodes probabilistes, à savoir leur coût important. Ceci s'explique par les nombreux paramètres, la difficulté à les estimer et la lente convergence du système.

Notons pour terminer, que les modèles retournés par ces algorithmes s'apparentent à une classification floue, formalisée par un ensemble de probabilités d'appartenance $\{p(C_h|x_i)\}_{h=1\ldots k}^{i=1\ldots n}$. Lorsque ces probabilités se rapprochent de leurs valeurs extrêmes 0 ou 1, les observations vérifient principalement l'un des k modèles, entraînant ainsi un schéma de clustering constitué de clusters bien séparés.

1.6.4 Le clustering par grilles

Le clustering par grilles procède par découpage de l'espace de représentation des données en un ensemble de cellules (*e.g* hyper-cubes, hyper-rectangles etc). De ce fait, ces méthodes visent principalement le traitement de données spatiales. Le résultat d'une telle méthode est une partition des données *via* une partition de l'espace de représentation des données. Les clusters formés correspondent à un ensemble de cellules denses et connectées.

La principale difficulté de ces méthodes concerne la recherche d'une taille appropriée pour les cellules construites (problème de granularité). De trop petites cellules conduiraient à un "sur-partitionnement", à l'inverse de trop grandes cellules entraîneraient un "sous-partitionnement".

Nous présentons, dans cette section, trois algorithmes de clustering utilisant un découpage en grilles : STING, WaveCluster et GIZMO. Ces trois méthodes résument les principales approches existantes dans ce domaine :
- construction d'une hiérarchie de grilles (STING),
- détection de limites entre zones à forte et à faible densités (WaveCluster),
- fusion des cellules de densités comparables et connectées (GIZMO).

L'algorithme STING (*STatistical INformation Grid-based method*) [183] propose une structure de grilles hiérarchique permettant une analyse rapide des données, dans le cadre d'une réponse à une requête utilisateur du type :

`Sélectionner les régions` R `telles que :`

- { R `contient au minimum 100 maisons par unité d'aire,`
- { `70% des maisons dans` R `ont un prix d'environ 400,000$,`
- { `l'aire totale de` R `est au moins de 100 unités avec une confiance de 90%.`

Cette structure hiérarchique est telle que la racine ($1^{ère}$ couche) correspond à une cellule (un rectangle dans le cas bidimensionnel) contenant l'ensemble des objets. Chaque nœud de la hiérarchie, autre que la racine ou une feuille, possède un nombre fixé[12] de nœuds fils pour lesquels chaque cellule correspondant est un quadrant de la cellule mère. Le découpage de l'espace est stoppé à un niveau de granularité impulsé par l'utilisateur. Nous proposons en figure 1.16 une représentation graphique de la structure hiérarchique générée par l'algorithme STING, dans le cas bidimensionnel.

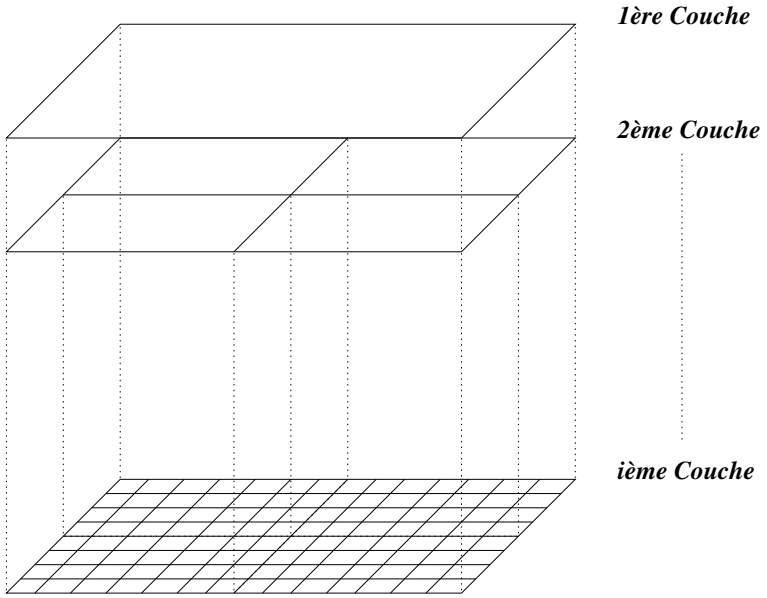


FIG. 1.16 – Structure hiérarchique 2D générée par STING.

Chaque cellule est caractérisée par un paramètre indépendant des attributs : n (nombre d'objets dans la cellule) et quatre paramètres dépendant des attributs : m (valeur moyenne), s (écart type), min (valeur minimum), max (valeur maximum) et $distrib$ (type de distribution). Les quatre derniers paramètres sont calculés pour chaque attribut. Le dernier paramètre ($distrib$) correspond au type de distribution éventuellement observée pour les objets de la cellule pour un attribut donné (*e.g* distributions normale, uniforme, exponentielle etc.). Cette recherche s'effectue généralement par une série d'hypothèses validées ou non par un test statistique tel que le test du χ^2.

Notons que ces paramètres sont calculés pour les feuilles de la hiérarchie puis transmis simplement de proche en proche aux nœuds supérieurs comme suit :

Soient (n, m, s, min, max) *les paramètres recherchés pour une cellule donnée, et* $\{(n_i, m_i,$

[12] La valeur par défaut est 4.

$s_i, min_i, max_i)\}$ *les paramètres connus pour les cellules filles correspondantes :*

$$n = \sum_i n_i \quad m = \frac{\sum_i m_i n_i}{n} \quad , \quad s = \sqrt{\frac{\sum_i (s_i^2 + m_i^2) n_i}{n} - m^2}$$

$$min = \min_i(min_i) \quad , \quad max = \max_i(max_i)$$

Le paramètre *dist* est également obtenu en fonction des distributions observées dans les cellules filles, le calcul de ce paramètre est cependant plus complexe (cf. [183]).

Une fois la structure hiérarchique construite et paramétrée, l'algorithme STING recherche dans cette structure, les cellules pertinentes pour une requête donnée, par une stratégie de recherche descendante. Il n'est cependant pas utile de débuter systématiquement à la racine mais plutôt dans une couche intermédiaire. Au niveau de cette couche, les cellules pertinentes sont recherchées (proportionnellement au nombre d'objets vérifiant la requête dans cette cellule) puis permettent de diriger et de limiter la recherche dans les couches inférieures jusqu'à la sélection d'un ensemble de cellules valides pour la requête.

L'algorithme STING permet donc de traiter de très grandes bases de données en proposant d'une part une structure hiérarchique de découpage de l'espace des données et d'autre part une méthode de parcours de cette structure aboutissant à la sélection d'un ensemble de cellules répondant à des caractéristiques exprimées sous forme d'une requête.

L'algorithme WaveCluster (*Wavelet-Based clustering*) [165] utilise les ondelettes - une méthode de transformation pour le traitement de signaux - sur les données synthétisées par un découpage de l'espace des attributs en grille [119]. L'algorithme (figure 1.17) procède en premier lieu à un découpage de l'espace des attributs (espace d-dimensionnel) tel que chaque dimension i est divisée en m_i intervalles. La grille construite possède alors $\prod_i m_i$ cellules, chaque objet étant affecté à une cellule.

La répartition des objets dans les cellules compose un signal d-dimensionnel, dont les parties à haute fréquence correspondent aux frontières des clusters tandis que les parties à basse fréquence permettent de distinguer les régions de l'espace où les objets sont concentrés.

L'utilisation de la méthode des ondelettes permet d'obtenir la décomposition appropriée du signal et de détecter ainsi la position des clusters. L'analyse du signal transformé passe par la recherche de cellules connexes (clusters) dans les différentes bandes de fréquence. Les objets sont finalement étiquetés en fonction des clusters obtenus par cette technique.

Notons qu'il est effectué, en général, plusieurs transformations[13] successives du signal, correspondant à différents niveaux de granularité.

L'algorithme WaveCluster se limite à des applications où les données multidimensionnelles sont décrites par des attributs numériques. La méthode est de complexité linéaire $O(n)$ sur le nombre d'objets dans le cas bidimensionnel mais cette complexité augmente de façon exponentielle avec la dimension de l'espace des attributs. Enfin, WaveCluster se distingue par sa capacité à gérer les outlier et à découvrir des clusters de formes variées.

Plutôt que d'utiliser le découpage en grilles dans le but de faciliter la recherche des contours des clusters, **l'algorithme GIZMO** [19] fusionne les cellules denses et connectées,

[13]Par défaut, WaveCluster effectue 3 transformations.

Algorithme WaveCluster : *Wavelet-Based Clustering*

Entrée : Un ensemble X de données multidimensionnelles,
Sortie : Une partition de X.

1. Découper l'espace des attributs et affecter chaque objet à une cellule,
2. Appliquer la transformation par ondelettes sur le signal multidimensionnel induit par les densités des cellules,
3. Rechercher les composants connectés (clusters) dans les différentes bandes de fréquence,
4. Affecter les objets aux clusters extraits.

FIG. 1.17 – L'algorithme WaveCluster.

susceptibles de former un cluster. GIZMO commence par partitionner l'espace des objets de façon similaire à l'étape de découpage utilisée dans WaveCluster. Ensuite, l'algorithme traite l'ensemble des cellules non vides (présence d'au moins un objet) ; s'appuyant sur les notions de *voisinage*[14] et de *connexion*[15] entre cellules, GIZMO détermine l'ensemble des composantes connexes (ensembles de cellules connectées). Chacune de ces composantes est alors considérée comme un cluster candidat et donne lieu à une analyse permettant de sélectionner les composantes sensiblement uniformes relativement à leur densité. Les composantes non retenues sont de nouveau analysées en les privant de leur cellule de plus faible densité.

Brézellec et Didier [19] montrent que l'algorithme GIZMO conduit à des résultats équivalents à la méthode WaveCluster. Les caractéristiques de cette dernière méthode sont transposables à GIZMO, notamment la découverte de clusters de formes variées.

Un inconvénient commun aux trois précédentes méthodes est leur faibles performances dans le cas de données décrites dans un espace à grande dimension. En effet la quantité de cellules augmente exponentiellement avec la dimension de l'espace des données. Un parcours simple de chaque cellule devient très coûteux dans cette situation. On serait alors amené à conclure que les méthodes de clustering basées sur un découpage de l'espace en grilles se limitent aux données de faible dimension. Cependant, une parade a été proposée par Agrawal et al. *via* **l'algorithme CLIQUE** (*CLustering In QUEst*) [4], utilisant les grilles pour traiter des données de grande dimension.

Cet algorithme repose sur l'idée que les objets ne sont pas distribués identiquement dans tous les sous-espaces de l'espace initial. Ainsi, ils peuvent être uniformément répartis dans certains sous-espaces ou au contraire s'organiser en régions de densités différentes dans d'autres. Ces derniers sous-espaces sont alors susceptibles d'aider à la détection des clusters.

La recherche des sous-espaces pertinents est la principale originalité de l'algorithme CLIQUE. Plutôt que de considérer la recherche des clusters dans un espace de grande dimension, CLIQUE traite plusieurs sous-problèmes, dans des espaces de dimension plus faible.

[14] Deux cellules sont voisines si elles se touchent sur la grille par une face ou un coin.

[15] Deux cellules sont connectées si elles sont voisines ou si il existe un chemin de cellules voisines permettant de les relier.

Après avoir découpé chaque dimension en intervalles réguliers (dont les paramètres doivent être spécifiés par l'utilisateur), CLIQUE s'appuie alors sur trois phases de traitement :

1. la recherche des sous-espaces susceptibles de contenir des clusters denses,
2. l'identification des clusters dans ces sous-espaces,
3. la construction d'une description (minimale) de ces clusters.

La première étape de recherche des sous-espaces intéressants se fait de manière ascendante : en commençant par sélectionner les sous-espaces de dimension 1 contenant des cellules (ici intervalles) denses[16], puis les combinaisons des sous-espaces sélectionnés formant des sous-espaces de dimension 2 susceptibles de contenir des cellules denses et ainsi de suite jusqu'à avoir généré tous les sous-espaces pertinents, relativement à la couverture de ses sous-espaces par des cellules denses. Cette méthode de recherche est basée sur la propriété de monotonie suivante : *"Si une cellule est dense dans un espace de dimension k, toute projection de cette cellule dans l'un des k sous-espaces de dimension $k-1$ est dense"*.

L'identification des clusters, dans chacun des sous-espaces pertinents, s'effectue par une stratégie de recherche semblable à celle utilisée dans l'algorithme GIZMO (recherche des composantes connexes dans un graphe formalisant les connexions entre cellules denses).

Enfin, Agrawal et al. proposent une stratégie pour générer une description minimale des clusters, en construisant des hyper-rectangles par généralisations successives initialisées par une cellule dense choisie au hasard[17].

L'algorithme CLIQUE présente donc l'avantage de permettre le traitement de données volumineuses dans des espaces de grande dimension. De plus, les descriptions proposées peuvent induire des intersections entre clusters, aboutissant ainsi à un pseudo-partitionnement des données.

Notons que l'ensemble des quatre algorithmes exposés (STING, WaveCluster, GIZMO et CLIQUE) utilisent la notion de densité. L'intérêt de cet apport est de ne pas limiter la recherche uniquement aux clusters constitués d'objets très similaires, mais au contraire, d'autoriser la formation de classes d'objets moins "proches" mais de densités homogènes. Les algorithmes fondés sur les densités font de cette notion le point central de leur processus, nous les présentons dans la section suivante.

1.6.5 Le clustering par densités

Les algorithmes présentés dans la section précédente utilisent, pour la plupart, la notion de "densité d'une cellule", définie relativement au nombre d'objets contenus dans cette cellule. Les algorithmes de clustering par densités se basent sur une notion similaire, complétée par d'autres concepts fondamentaux tels que le voisinage d'un objet, l'objet noyau, l'accessibilité ou la connexion entre objets. Nous proposons d'abord de définir chacune de ces notions avant de présenter les algorithmes qui en découlent.

Définition 1.16. *Soit un objet $x_i \in X$; le* **voisinage** *$\mathcal{N}_\epsilon(x_i)$ de x_i (de rayon ϵ) est défini par l'ensemble des points de X, distants d'au plus ϵ de x_i :*

$$\mathcal{N}_\epsilon(x_i) = \{x_j \in X | d(x_i, x_j) \leq \epsilon\}$$

[16] La densité d'une cellule est définie relativement à un seuil fixé par l'utilisateur : % d'objets $> \tau$ dans la cellule.

[17] Cette stratégie s'apparente à la méthode de l'étoile proposée par Michalski et al.[84]

Définition 1.17. *Soient un objet $x_i \in X$, ϵ et M deux paramètres fixés ; x_i est un* **objet noyau** *dans X si et seulement si le voisinage de x_i contient au moins M objets :*

$$\textit{noyau}(x_i) \Leftrightarrow |\mathcal{N}_\epsilon(x_i)| \geq M$$

Définition 1.18. *Soient deux objets $x_i, x_j \in X$, ϵ et M deux paramètres fixés ; x_i est* **directement** *d-***accessible**[18] *par x_j, si et seulement si les deux propriétés suivantes sont vérifiées :*

i) $x_i \in \mathcal{N}_\epsilon(x_j)$,

ii) $|\mathcal{N}_\epsilon(x_j)| \geq M$ *(x_j est un objet noyau).*

Ainsi x_i est *directement d-accessible* par x_j si le voisinage de x_j contient au minimum M objets, dont x_i. Notons alors que cette relation n'est ni symétrique, ni transitive. La relation suivante n'est toujours pas symétrique mais transitive :

Définition 1.19. *Soient deux objets $x_i, x_j \in X$, ϵ et M deux paramètres fixés ; x_i est* *d-***accessible** *par x_j si il existe une chaîne $x'_1, \ldots, x'_m$ dans X telle que $x'_1 = x_j$, $x'_m = x_i$ et $\forall i = 1, \ldots, m-1$, x'_{i+1} est directement d-accessible par x'_i.*

Définition 1.20. *Soient deux objets $x_i, x_j \in X$, ϵ et M deux paramètres fixés ; x_i est* *d-***connecté**[19] *à x_j si il existe un objet $x_k \in X$ tel que x_i et x_j sont tous deux d-accessibles par x_k.*

Cette dernière notion correspond cette fois à une relation symétrique mais non transitive.

Sur un principe comparable aux algorithmes de ré-allocations dynamiques, les méthodes basées sur la densité cherchent à découvrir des clusters pour lesquels tous les objets sont en relation avec un même objet central :

- dans le cas des méthodes par ré-allocations, les objets centraux sont les objets représentatifs (centroïdes, médoïdes etc.) et la relation est basée sur la similarité,
- pour les méthodes basées sur la densité, les objets centraux sont des noyaux par lesquels les autres objets sont d-accessibles.

Définition 1.21. *Soient X l'ensemble des objets, ϵ et M deux paramètres fixés. Un* *d-***cluster** *$C \subset X$ est un ensemble non vide dans X vérifiant les propriétés suivantes :*

i) Maximalité : $\forall x_i, x_j \in X$: *si $x_i \in C$ et x_j est d-accessible par x_i, alors $x_j \in C$,*

ii) Connectivité : $\forall x_i, x_j \in C$: *x_i est d-connecté à x_j.*

Lemme 1.1. *Soient x_i un objet noyau dans X et O l'ensemble des objets d-accessibles par x_i : alors O est un d-cluster.*

Lemme 1.2. *Soient C un d-cluster et x_i un noyau dans C : alors C est l'ensemble des objets d-accessibles par x_i.*

Les deux lemmes précédents, dont les preuves sont présentées dans [55], permettent d'orienter la recherche d'un d-cluster. En effet, la construction d'un tel cluster revient à rechercher d'abord un objet noyau puis à agglomérer, autour de ce noyau, tous les objets d-accessibles par ce noyau. L'algorithme DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) [56] résume assez bien cette stratégie de recherche. Les grandes étapes de cet algorithme sont présentées en figure 1.18.

[18] En Anglais *directly density-reachable.*

[19] En Anglais *density-connected.*

DBSCAN : *Density-Based Spatial Clustering of Applications with Noise*

Entrées : Un ensemble X de n objets, ϵ et M deux paramètres fixés,
Sortie : Une partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ de X en k d-clusters.

1. Initialisation $id = 1$ et $C_{id} = \emptyset$,
2. Pour i allant de 1 à n :
    3. Si x_i n'est pas un noyau ou si $x_i \in \bigcup_{j=1\ldots id} C_j$, alors retourner à l'étape 2,
    4. `construire-cluster`(x_i,X,C_{id},ϵ,M),
    5. $id = id + 1$ et $C_{id} = \emptyset$,
6. Retourner l'ensemble des d-clusters : $C_1, \ldots, C_{id-1}$.

FIG. 1.18 – L'algorithme DBSCAN.

La procédure `construire-cluster` commence par ajouter, dans le cluster en cours d'élaboration, tous les objets dans le voisinage du noyau détecté x_i. Pour chaque objet ajouté on teste s'il s'agit d'un noyau, le cas échéant le cluster est étendu à ses objets d-connectés. Cette procédure récursive conduit à un d-cluster tel que nous l'avons défini, vérifiant les propriétés de :

Maximalité : Tous les objets d-accessibles par un objet du cluster ont été ajoutés de proche en proche,

Connectivité : Quelque soient les deux objets du cluster, ils sont nécessairement tous les deux d-accessibles par le noyau initial x_i, donc d-connectés entre eux.

L'algorithme DBSCAN est de complexité $O(n \log n)$ ce qui en fait une méthode assez peu coûteuse. De plus, les clusters obtenus peuvent être de formes variées. Enfin, DBSCAN distingue bien les objets bruités en déclarant comme *outliers* les objets qui ne sont d-connectés à aucun autre objet. Cependant l'algorithme présente un inconvénient majeur : le choix des paramètres ϵ et M. Même si les auteurs de l'algorithme proposent une heuristique pour déterminer automatiquement ces paramètres, ce choix reste difficile en pratique. Les données ne sont généralement pas distribuées identiquement et ces paramètres devraient pouvoir varier suivant les régions de l'espace.

Les mêmes auteurs proposent alors l'algorithme OPTICS (*Ordering Points To Identify the Clustering Structure*) [5]. OPTICS définit un ordre sur les objets qui peut ensuite être utilisé par DBSCAN, dans la phase d'expansion des clusters. Cet ordre permet ainsi d'envisager plusieurs niveaux de densités, caractérisés par plusieurs valeurs du paramètre ϵ. Cet ordre est défini à l'aide de deux nouvelles notions : la *distance au noyau* et la *distance d'accessibilité*.

L'algorithme DBCLASD (*Distribution-Based Clustering of Large Spatial Databases*) [187] propose, quant à lui, une approche distributionnelle pour gérer ce problème de variation des densités locales. DBCLASD recherche la distribution de probabilité d'une variable aléatoire correspondant à la distance d'un objet à son plus proche voisin (*dppv*). Sous les hypothèses que les objets sont distribués uniformément à l'intérieur d'un même cluster et que chaque cluster possède sa propre "échelle" de *dppv*, DBCLASD découvre des clusters de différentes densités et de formes variées.

Enfin, l'algorithme GDBSCAN (*Generalized DBSCAN*) [160] a été proposé comme extension à DBSCAN pour traiter des données spatiales plus complexes telles que les polygones.

L'algorithme DBSCAN et les variantes présentées précédemment pourraient aisément produire un pseudo-partitionnement de l'ensemble des données. En effet, un objet qui n'est pas un noyau peut être d-accessible par plusieurs objets de différents clusters. Dans DBSCAN cette situation est évitée en interdisant l'ajout dans un cluster, d'un objet déjà ajouté à un précédent cluster. La suppression de cette interdiction laisserait apparaître des intersections entre clusters, même si ces situations semblent "rares".

Un partitionnement flou peut également être envisagé par une méthode de clustering basée sur la densité. L'algorithme DENCLUE (*DENsity-based CLUstEring*) [82] recherche un ensemble de points *"density-attractors"* qui s'apparentent aux centres de gravité dans l'algorithme des k-moyennes flou. Ces points sont, en fait, les maxima locaux d'une fonction de densité globale rendant compte de l'influence de l'ensemble des objets en tout point de l'espace. Ces influences sont modélisées par des *fonctions d'influence* qui mesurent, pour chaque objet, son impact dans son voisinage. Ce sont ces dernières fonctions qui font intervenir la notion de densité par l'utilisation des voisinages. Cependant, DENCLUE n'est pas destiné à la construction de clusters flous et souffre également d'un excès de paramètres.

1.6.6 Le clustering conceptuel

Le clustering conceptuel a été introduit au début des années 1980 par [124]. Cette approche du clustering est alors présentée comme une manière de découvrir des schémas (clusters) "compréhensibles" à partir des données. Plutôt que de définir une mesure de similarité puis d'organiser les objets de façon à minimiser les inerties intra-clusters et maximiser l'inertie inter-clusters, Michalski propose de générer une structure (hiérarchique) de concepts. Dans ce type de structure, chaque concept se définit à la fois en extension (ensemble des objets placés dans ce concept) et en intension (règles descriptives sur les objets). La définition en intension permet une caractérisation des clusters (ou concepts) et fournit alors à l'utilisateur une explication compréhensible de ces concepts.

Plusieurs systèmes de clustering conceptuel ont été proposés, afin de traiter des données plus ou moins complexes et de construire différents types de structures conceptuelles. On note alors les premiers algorithmes tels que CLUSTER/2 [125] modifié dans CLUSTER/S [177] pour traiter des données à domaines de valeurs davantage structurés, comparativement aux descriptions sous forme attribut/valeur. De la même façon, l'algorithme incrémental COBWEB [60] a été adapté dans CLASSIT [66] pour traiter des données décrites par des attributs numériques. Contrairement à ces dernières méthodes, qui génèrent chacune un arbre hiérarchique, d'autres approches consistent à organiser les concepts dans des graphes conceptuels, s'apparentant davantage à des treillis de Galois. Cet autre type de structure est notamment étudié dans les travaux de [184, 174].

Nous avons choisi de présenter, dans cette section, l'algorithme COBWEB, afin d'illustrer le fonctionnement des méthodes de clustering conceptuel. Cet algorithme est très utilisé et se présente souvent comme une référence dans ce domaine.

COBWEB est un système incrémental de clustering conceptuel hiérarchique. Le processus de formation de concepts a pour objectif de construire, à partir d'un ensemble de

données et de leur description, une hiérarchie de concepts telles que nous la présentons en figure 1.19.

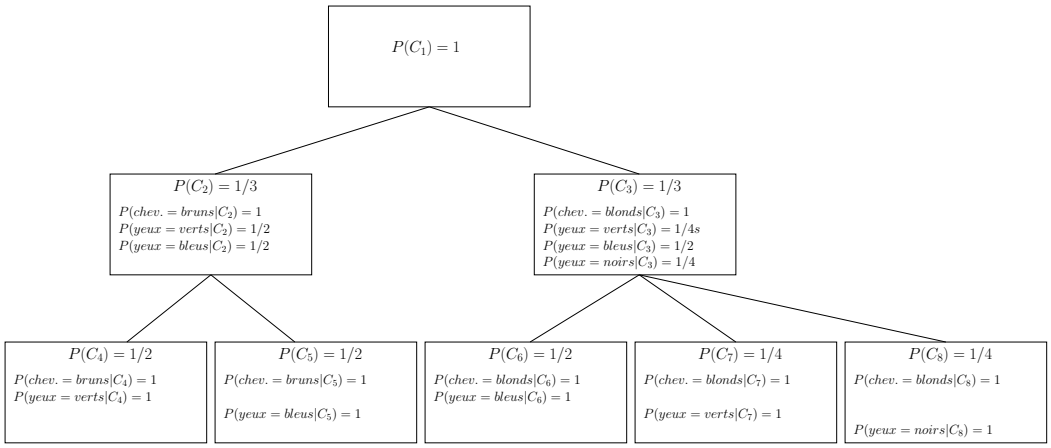


FIG. 1.19 – Exemple d'une hiérarchie de concepts générée par COBWEB.

A chaque concept de la hiérarchie est associée une définition générique des objets constituant la classe. Par exemple, dans la figure 1.19, la classe C_2 contient 1/3 des individus de l'ensemble initial et dans C_2 tous les individus ont les cheveux bruns et la moitié ont les yeux verts.

L'algorithme COBWEB s'inscrit dans la problématique générale du clustering puisqu'il s'agit d'optimiser un critère global basé sur les deux notions de :

- similarité intra-classe,
- dissimilarité inter-classes.

Le critère *CU* (*Category Utility*) [69] est alors utilisé dans COBWEB pour évaluer la qualité globale d'un schéma de clustering :

$$CU(\{C_1, \ldots, C_t\}) = \frac{1}{t} \sum_{k=1}^{t} p(C_k)[\sum_i \sum_j p(A_i = V_{i,j}|C_k)^2 - \sum_i \sum_j p(A_i = V_{i,j})^2] \quad (1.1)$$

Dans cette dernière définition, $\{A_1, \ldots, A_p\}$ désigne l'ensemble des attributs décrivant les objets et $\{V_{i,1}, \ldots, V_{i,n_i}\}$ le domaine de valeurs de l'attribut A_i. On retrouve dans ce critère les probabilités conditionnelles $p(A_i = V_{i,j}|C_k)$ et $p(C_k|A_i = V_{i,j})$[20], formalisant respectivement la similarité intra-classe et la dissimilarité inter-classes. Si un couple attribut/valeur $(A_i = V_{i,j})$ est indépendant d'une classe C_k, alors $p(A_i = V_{i,j}|C_k) = p(A_i = V_{i,j})$ et $p(A_i = V_{i,j}|C_k)^2 - p(A_i = V_{i,j})^2 = 0$. En supposant que cela soit vrai pour tous les couples attribut/valeur, le schéma de concepts proposé est indépendant des descriptions et le critère *CU* atteint son minimum 0. En revanche, ce critère atteint son maximum pour le meilleur schéma conceptuel.

La stratégie de construction de la structure hiérarchique consiste à incorporer les objets de façon incrémentale dans la structure. Chaque nouvel objet est introduit par la racine puis parcourt l'arbre en appliquant successivement l'un des opérateurs suivants :

Incorporer l'objet dans le cluster,

[20]Ce terme peut être retrouvé par la règle de Bayes.

COBWEB : Algorithme incrémental de classification conceptuelle hiérarchique.

Entrées : Le nœud courant N de la hiérarchie de concepts et x_i une nouvelle instance à incorporer,
Sortie : Une hiérarchie de concepts classifiant l'instance x_i.

Procédure `COBWEB`(N,x_i)

Si N est un nœud terminal (feuille) alors :
`créer-nouveaux-nœuds-terminaux`(N,x_i)
`incorporer`(N,x_i)

Sinon `incorporer`(N,x_i)
Pour chaque nœud fils C de N :
Calculer le score d'intégration de x_i dans C
Soit P le nœud avec le plus haut score (S_I),
Soit R le nœud avec le deuxième plus haut score,
Soit S_C le score correspondant à la création d'un nouveau nœud $Q = \{x_i\}$,
Soit S_F le score correspondant à la fusion des nœuds P et R,
Soit S_D le score correspondant à la division du nœud P (remplacé par ses fils),

Si S_I est le meilleur score alors : `COBWEB`(P,x_i),
Sinon si S_C est le meilleur score alors : initialiser Q relativement à x_i,
Sinon si S_F est le meilleur score alors : O = `fusionner`(P,R,N) et `COBWEB`(O, x_i),
Sinon si S_D est le meilleur score alors : `diviser`(P,N) et `COBWEB`(N, x_i).

FIG. 1.20 – L'algorithme COBWEB.

Créer un nouveau cluster,

Fusionner deux clusters en un seul,

Diviser un cluster en plusieurs clusters.

Ces quatre opérations sont détaillées en figure 1.21 et viennent en complément de l'algorithme général COBWEB reporté en figure 1.20, selon le formalisme proposé par [67].

COBWEB présente plusieurs limites parmi lesquelles : la difficulté à traiter les attributs numériques, l'influence de l'ordre dans lequel les objets sont intégrés à la hiérarchie ou encore le stockage coûteux de l'ensemble des instances dans cette hiérarchie.

Les algorithmes de clustering conceptuel génèrent, pour la plupart, une structure hiérarchique de concepts (graphe ou arbre). De même que les hiérarchies peuvent être généralisées en pseudo-hiérarchies ou pyramides (cf. 1.6.1), un arbre, obtenu par exemple avec COBWEB, peut être généralisé en une pyramide conceptuelle. Dans ce type de structure, à chaque nœud est associée une définition en intension, et à un même niveau dans la pyramide, chaque concept peut être partiellement recouvert par au plus deux autres

fonction `incorporer`(N,x_i)

Mettre à jour les probabilités de N,
Pour chaque attribut A de x_i :
    Pour chaque valeur V de A :
        Mettre à jour la probabilité $p(V|N)$.

fonction `créer-nouveaux-nœuds-terminaux`(N,x_i)

Créer un nouveau nœud fils M au nœud N,
Initialiser les probabilités de M à partir des probabilités de N,
Créer un nouveau fils O au nœud N,
Initialiser les probabilités de O à relativement à x_i.

fonction `fusionner`(P, R, N)

Créer un nouveau nœud fils O au nœud N,
Mettre à jour les probabilités de O par moyenne des probabilités de P et R,
Supprimer les nœuds fils P et R de N,
Ajouter P et R comme nœuds fils du nœud O,
Retourner O.

fonction `Diviser`(P, N)

Supprimer le nœud fils P de N,
Ajouter chaque nœud fils de P comme nouveau nœud fils de N.

FIG. 1.21 – Fonctions auxiliaires de COBWEB.

concepts. Ces travaux plutôt récents sont présentés dans [20] et ont donné lieu au système SODAS [47].

1.6.7 Quelques autres approches de clustering

Nous avons cité jusqu'ici, les approches incontournables permettant de constituer des classes d'objets, à partir de leur description ou seulement d'une matrice de (dis)similarité, dans un contexte non-supervisé. Ces algorithmes hiérarchiques ou de partitionnement, constituent une synthèse assez large de ce domaine qu'est la reconnaissance de formes. Cependant, cet état de l'art ne saurait être complet sans mentionner certaines techniques ou formalismes provenant d'autres domaines de l'apprentissage ou plus généralement de l'informatique, tels que les réseaux de neurones artificiels ou la théorie des graphes.

Les réseaux de neurones artificiels, inspirés des réseaux de neurones biologiques [81], sont célèbres pour leur utilisation en classification (supervisée). Cependant, les caractéristiques de ces réseaux ont également été adaptées dans un cadre non-supervisé [13], notamment l'aspect parallélisation du processus, permettant ainsi le traitement de grandes bases de données. L'algorithme le plus connus est la méthode SOM, ou cartes auto-organisatrices de Kohonen (*Self-Organizing Map*) [101]. Il s'agit d'un réseau monocouche, où chaque objet d'entrée (neurone d'entrée) est associé à un neurone de sortie par projection. Ces "neurones de sortie" sont en fait les représentants des classes, en nombre fixé *a priori*, et organisés généralement sur une grille 2D. Cet algorithme procède de façon incrémentale : chaque vecteur d'entrée engendre une compétition entre neurones pour déterminer le neurone vainqueur (le plus proche), induisant alors la mise à jour des poids "synaptiques" du vainqueur et de son environnement proche. Malgré l'avantage considérable qu'apporte la visualisation du résultat dans un plan[21], les cartes auto-organisatrices présentent quelques limites importantes : la possibilité de traiter des vecteurs numériques uniquement, la dépendance du résultat avec l'initialisation du modèle et les paramètres de l'apprentissage (nombre d'itérations, voisinages d'un nœud vainqueur, etc.).

Le clustering basé sur les graphes est apparu dans les années 80. Cette approche est fondée sur le formalisme théorique des graphes, constitués de sommets et d'arêtes éventuellement valuées.

Il existe différentes stratégies pour générer ce type de graphes, appelés "graphes de proximités" [57] ; nous définissons dans ce qui suit les principales familles de graphes de proximités. Dans ces définitions, les graphes ont pour ensemble de sommets, l'ensemble des objets de X, sur lequel une mesure de distance d est définie.

Définition 1.22. *Le graphe du plus proche voisin $NNG(X)$ (Nearest Neighbor Graph [139]) est défini par l'ensemble X de sommets et l'ensemble V d'arêtes tel que chaque objet $x_i \in X$ est relié à son plus proche voisin dans X relativement à d.*

Définition 1.23. *L'arbre minimum de recouvrement $MST(X)$ (Minimum Spanning Tree [190]) est défini par l'ensemble X de sommets et l'ensemble V d'arêtes tel que $\forall (x_i, x_j) \in X \times X$ il existe un chemin dans $MST(X)$ de x_i à x_j et $\sum_{(x_i,x_j)\in V} d(x_i, x_j)$ est minimale.*

Définition 1.24. *Le graphe de voisinage relatif $RNG(X)$ (Relative Neighborhood Graph [178]) est défini par l'ensemble X de sommets et l'ensemble V d'arêtes tel qu'il existe une*

[21] La proximité des classes sur la grille est révélatrice de la proximité entre leurs objets.

arête entre deux objets x_i et x_j de X si et seulement si

$$\forall x_k \in X \setminus \{x_i, x_j\}, d(x_i, x_j) \leq \max\{d(x_i, x_k), d(x_j, x_k)\}$$

Définition 1.25. *Le graphe de Gabriel $GG(X)$ (Gabriel Graph [121]) est défini par l'ensemble X de sommets et l'ensemble V d'arêtes tel qu'il existe une arête entre deux objets x_i et x_j de X si et seulement si le cercle de diamètre $\overline{x_i x_j}$[22] ne contient aucun autre objet de X :*

$$\forall x_k \in X \setminus \{x_i, x_j\}, d(x_i, x_j)^2 < d(x_i, x_k)^2 + d(x_j, x_k)^2$$

Définition 1.26. *La triangulation de Delaunay $DT(X)$ (Delaunay Triangulation [108]) est définie par X l'ensemble de sommets et V l'ensemble d'arêtes tel qu'il existe une arête entre deux objets x_i et x_j si les cellules de Voronoï associées sont adjacentes*[23].

Propriété 1.6. *Les familles de graphes définies précédemment vérifient, dans un espace Euclidien, les relations d'inclusion suivantes (cf. [92]) :*

$$NNG \subseteq MST \subseteq RNG \subseteq GG \subseteq DT$$

Partant de l'un des graphes précédents, il s'agit alors de diviser l'ensemble des sommets du graphe, en plusieurs groupes "naturels", par rapport aux relations entre objets, caractérisées par les arêtes. On distingue deux approches majeures pour effectuer ce partitionnement.

La première approche fixe un nombre de clusters (k) et choisit un critère (*e.g.* nombre d'arêtes entre les clusters). L'algorithme de clustering recherche alors une partition du graphe en k ensembles équilibrés (homogènes en taille) de sommets et tels que le nombre d'arêtes inter-clusters soit minimal. La complexité d'une telle méthode augmente de façon exponentielle avec le paramètre k, c'est pourquoi la plupart des algorithmes proposent une approche récursive de partitionnement en 2 telle que $k = 2^p$ [141].

La seconde approche propose de choisir un critère (*e.g.* le diamètre maximum des clusters) puis d'effectuer le partitionnement de façon à obtenir le nombre optimal de clusters satisfaisant à ce critère [18]. Il s'agira, par exemple, de partitionner le graphe en un minimum de clusters de diamètre maximum fixé.

La représentation des données à l'aide de graphes et le clustering de graphes sont particulièrement adaptés à certaines applications telles que la recherche de communautés d'intérêt dans des réseaux sociaux ou le regroupement de pages web. Ces situations nécessitent de traiter des volumes importants de données ; les nombreuses heuristiques d'approximation issues des recherches fondamentales en théorie des graphes prennent alors une place importante dans ces applications.

1.7 Techniques d'évaluation du clustering

L'évaluation des résultats d'un clustering est un problème majeur, qui renvoie à la question première : qu'est-ce qu'un bon schéma de clustering ? Cette problématique est plutôt bien synthétisée dans les récents travaux de M. Halkidi [77, 76]. Trois cheminements sont alors envisagés pour évaluer ou comparer entre eux des schémas de clustering :

[22] Aussi appelé "cercle d'influence" de x_i et x_j.

[23] La cellule de Voronoï associée à un objet x_i correspond à l'ensemble des points de l'espace plus "proches" de x_i que de tout autre point de X.

L'évaluation externe : il s'agit de confronter un schéma avec une classification prédéfinie. L'évaluation porte donc sur l'adéquation entre le schéma obtenu et une connaissance "externe" sur les données (schéma attendu).

L'évaluation interne : ce type d'évaluation n'utilise pas de connaissances externes mais uniquement les données d'entrées (matrice de (dis)similarité, descriptions des données etc.) comme référence. Ainsi, par exemple, parmi plusieurs schémas, le meilleur sera celui qui conserve un maximum d'information relativement à l'information contenue dans la matrice de (dis)similarité.

L'évaluation relative : cette dernière stratégie d'évaluation porte généralement sur les deux principaux critères de dispersions intra-clusters (à minimiser) et inter-clusters (à maximiser). L'évaluation relative est souvent utilisée pour comparer plusieurs schémas obtenus par une même méthode avec différents paramétrages. Ceci permet alors de sélectionner les paramètres optimaux pour un algorithme, étant donné un ensemble de données.

Afin d'illustrer ces trois stratégies d'évaluation, nous choisissons de présenter l'indice Γ (aussi appelé "statistique de Huberts"), qui propose trois variantes correspondant chacune à l'une des trois stratégies mentionnées. Considérons l'ensemble $X = \{x_1, \ldots, x_n\}$ des objets à traiter et $M = \frac{n(n-1)}{2}$ le nombre de paires possibles dans X : $\{(x_i, x_j)\}_{i \neq j}$. L'indice Γ s'énonce alors comme suit :

$$\Gamma = \frac{1}{M} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} U(i,j).V(i,j)$$

Γ *externe* **:** dans ce cas $U(i,j)$ et $V(i,j)$ correspondent aux distances entres les clusters contenant les objets x_i et x_j, respectivement dans le schéma obtenu et dans la classification issue des connaissances externes (classification prédéfinie). L'optimum est atteint pour deux schémas identiques, caractérisés par $\Gamma = \frac{1}{M} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} V(i,j)^2$.

Γ *interne* **:** cet indice permet d'évaluer l'adéquation entre un schéma $\mathcal{C}$ et la matrice de dissimilarités D dont il est issu. Pour cela on pose $U(i,j) = d(x_i, x_j)$ (dissimilarité entre les deux objets) et

$$V(i,j) = \begin{cases} 1 & \textit{si } x_i \textit{ et } x_j \textit{ appartiennent à des clusters différents dans } \mathcal{C}, \\ 0 & \textit{sinon.} \end{cases}$$

On notera que le maximum est atteint pour le schéma $\mathcal{C} = \{\{x_1\}, \ldots, \{x_n\}\}$ (chaque cluster est constitué d'un unique objet). En revanche, pour un nombre fixe de clusters, l'optimum (maximum) caractérise un schéma constitué de clusters bien séparés relativement à la matrice D.

Γ *relatif* **:** dans ce dernier cas $U(i,j)$ correspond toujours à la distance entre les deux objets x_i et x_j tandis que $V(i,j)$ mesure la distance entre les clusters contenant respectivement x_i et x_j. Pour deux objets appartenant à un même cluster, le produit est nul ($V(i,j) = 0$). On cherche alors le schéma qui maximise ce critère, révélant ainsi des clusters compacts et séparés.

Nous donnons ensuite un bref aperçu des indices complémentaires à l'indice Γ, pouvant être utilisés dans chacune des trois stratégies d'évaluation. Nous terminons par une présentation de quelques indices destinés à évaluer des schémas flous et discutons des adaptations possibles pour le traitement de schémas avec recouvrements. Ces indices sont, pour la plupart, définis formellement en Annexe A.

1.7.1 Critères externes

D'autres mesures, complémentaires à l'indice Γ, ont été proposées pour comparer l'adéquation entre un schéma de clustering $\mathcal{C}$ et une classification préétablie $\mathcal{P}$. Par exemple des mesures de comptage, similaires à celles utilisées pour définir la proximité entre deux objets décrits par des attributs symboliques (cf. 1.4). Les indices de *Rand*, de *Jaccard* ou encore de *Fowlkes et Mallows* permettent de mesurer le taux de liaisons ou non-liaisons correctes dans le schéma à évaluer. Notons :

- a le nombre de paires (x_i, x_j) telles que x_i et x_j se retrouvent dans une même classe dans $\mathcal{C}$ et dans $\mathcal{P}$ (liaison correcte),
- b le nombre de paires (x_i, x_j) telles que x_i et x_j se retrouvent dans une même classe dans $\mathcal{C}$ mais non dans $\mathcal{P}$ (liaison incorrecte),
- c le nombre de paires (x_i, x_j) telles que x_i et x_j se retrouvent dans une même classe dans $\mathcal{P}$ mais non dans $\mathcal{C}$ (non-liaison incorrecte),
- d le nombre de paires (x_i, x_j) telles que x_i et x_j ne se retrouvent dans une même classe ni dans $\mathcal{C}$ ni dans $\mathcal{P}$ (non-liaison correcte).

Ces trois indices notés respectivement R, J et FM se définissent ainsi :

$$R(\mathcal{C},\mathcal{P}) = \frac{a+d}{a+b+c+d} \;;\; J(\mathcal{C},\mathcal{P}) = \frac{a}{a+b+c} \;;\; FM(\mathcal{C},\mathcal{P}) = \sqrt{\frac{a}{a+b}\,\frac{a}{a+c}}$$

La principale différence entre les indices de *Rand* et de *Jaccard* est la prise en compte, ou non, des non-liaisons correctes.

Une dernière mesure d'évaluation concerne la pureté des clusters obtenus dans $\mathcal{C}$ relativement à $\mathcal{P}$. Cette mesure n'est, bien sûr, pas indépendante des indices déjà mentionnés précédemment. La pureté d'un ensemble de clusters peut s'évaluer par la mesure d'entropie telle que nous la définissons ici :

$$E(\mathcal{C},\mathcal{P}) = \sum_{i=1}^{k_\mathcal{C}} \frac{|C_i|.\varphi(C_i)}{n} \quad \text{avec} \quad \varphi(C_i) = -\sum_{j=1}^{k_\mathcal{P}} p_j.\log p_j$$

Dans cette dernière définition, $k_\mathcal{C}$ et $k_\mathcal{P}$ désignent le nombre de clusters respectivement dans $\mathcal{C}$ et dans $\mathcal{P}$; p_j correspond à la proportion (dans C_i de $\mathcal{C}$) d'objets appartenant à la classe j de $\mathcal{P}$. Le schéma d'entropie minimale, correspond au schéma de pureté maximum relativement à la classification attendue.

Le plus souvent, il n'existe pas de classification connue à l'avance et pouvant servir de référence. Quand bien même une telle connaissance existe, cette dernière peut être fondée sur des informations qui ne se retrouvent ni dans la description proposée des données ni, indirectement, dans la matrice de (dis)similarité calculée à partir de ces descriptions. Il est donc nécessaire de pouvoir évaluer un schéma de façon autonome, c'est-à-dire en utilisant uniquement les informations dont l'algorithme dispose. Ce sont les critères d'évaluation interne ou relative qui sont alors utilisés.

1.7.2 Critères internes

Comparativement aux deux autres approches, il existe peu de mesures d'évaluation interne. Dans le cas d'un partitionnement, c'est l'indice Γ (*interne*) qui est généralement utilisé. En revanche, lorsqu'il s'agit de l'évaluation d'une méthode de clustering hiérarchique, on utilise de coefficient de corrélation cophénétique (CPCC). Pour cela on construit la

matrice cophénétique[24] (P_C) de la hiérarchie produite par l'algorithme de clustering, que l'on compare à la matrice de proximité (P). Pour $M = \frac{n(n-1)}{2}$, μ_P et μ_C les moyennes des matrices respectives P et P_C, le coefficient CPCC est défini par :

$$CPCC(P_C, P) = \frac{\frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P(i,j).P_C(i,j) - \mu_P\mu_C}{\sqrt{\left[\frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P(i,j)^2 - \mu_P^2\right]\left[\frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P_C(i,j)^2 - \mu_C^2\right]}}$$

Cet indice prend ses valeurs entre -1 et 1 ; une valeur proche de 0 est significative d'une forte similarité entre les deux matrices et donc d'un bon schéma de clustering.

1.7.3 Critères relatifs

Ce type d'évaluation, comme nous l'avons déjà mentionné, permet de choisir les paramètres optimaux étant donnée une méthode de clustering et un ensemble de données. L'un des principaux paramètres que l'on peut chercher à apprendre est le nombre k de clusters à construire. La méthode traditionnelle consiste alors à faire varier un seul paramètre à la fois et à observer la qualité des schémas obtenus ; le paramétrage optimal est déterminé par une "cassure", significative d'un meilleur schéma, dans le tracé de l'indice utilisé.

Plusieurs types d'indices existent, en fonction des sources que l'on souhaite utiliser (descriptions ou matrice de (dis)similarité), et du type de schéma produit par la méthode (hiérarchie ou partitionnement).

Ces indices sont généralement basés sur l'évaluation de la dispersion intra-clusters et/ou inter-clusters. Par exemple l'un des premiers indices proposé est celui de *Dunn* [51] :

$$D(\mathcal{C}) = \min_{i=1\ldots t}\min_{j\neq i}\left(\frac{\min_{x\in C_i, y\in C_j} d(x,y)}{\max_{k=1\ldots t} diam(C_i)}\right)$$

L'indice de *Dunn* combine les deux précédentes notions en considérant à la fois le diamètre des clusters (dénominateur) et la séparation entre deux clusters (numérateurs). Quelques années plus tard, *Davies* et *Bouldin* [36] proposent un nouvel indice reprenant ces deux critères de dispersion intra/inter clusters :

$$DB(\mathcal{C}) = \frac{1}{t}\sum_{k=1}^{t}\max_{j\neq k}\frac{V(C_k)+V(C_j)}{d(x_k^*, x_j^*)}$$

où $V(C_i)$ désigne la variance d'un cluster C_i et x_i^* le représentant (centroïde ou médoïde) de C_i.

Ces deux indices se basent uniquement sur la matrice de (dis)similarité et peuvent être utilisés aussi bien pour comparer des partitions ou des hiérarchies.

Pour terminer la présentation des critères relatifs, on peut mentionner d'autres indices aidant à choisir à quel niveau, dans une hiérarchie, il est souhaitable d'extraire un schéma

[24] La matrice cophénétique est définie par les valeurs $P_C(x_i, x_j)$ (ultramétriques), où chaque valeur correspond au premier niveau (similarité) de la hiérarchie, pour lequel les deux objets se retrouvent dans un même cluster.

de clustering : RMSSTD (*Root-mean square deviation*), SPR (*Semi-partial R-squared*) ou encore RS (*R-squared*). Ces indices utilisent principalement les descriptions des données plutôt qu'une matrice de similarité.

1.7.4 Critères d'évaluation pour clustering flou

Dans le cas de schémas flous, il s'agit d'adapter les critères précédents en intégrant les valeurs d'appartenance ($\{u_h(x_i)\}_{i=1\ldots n, h=1\ldots t}$) des objets aux clusters à évaluer. L'indice de séparation ($S(\mathcal{C})$) se place comme une variante floue de l'indice de *Davies-Boulin* :

$$S(\mathcal{C}) = \frac{\sum_{h=1}^{t}\sum_{i=1}^{n} u_h^2(x_i) d(x_i, x_h^*)^2}{n. \min_{h\neq l} d(x_h^*, x_l^*)^2}$$

Dans cette définition, le numérateur correspond à la somme des inerties intra-clusters (floue) tandis que le dénominateur évalue la séparation inter-clusters. Un bon schéma de clustering doit donc minimiser cet indice de séparation. Enfin les indices de *Xie* et *Beni* [186] ou de *Fukuyama* et *Sugeno* [63] présentent des stratégies d'évaluation assez similaires à l'indice de séparation. Il existe de nombreux autres critères du même genre, utilisant cette fois les descriptions numériques des données.

Nous terminerons cette section en présentant le coefficient de partition ($PC(\mathcal{C})$) que nous aurons l'occasion d'utiliser par la suite. Plutôt que d'évaluer un schéma (flou) de façon qualitative, ce coefficient est utilisé comme indicateur de la proportion de "flou" dans la partition :

$$PC(\mathcal{C}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{h=1}^{t} u_h^2(x_i)$$

La partition la plus floue possible se caractérise par des valeurs d'appartenance toutes identiques et égales à $1/t$; dans ce cas le coefficient est minimum et vaut $1/t$. Inversement, lorsque la partition évaluée est stricte (chaque objet possède l'une de ses valeurs d'appartenance égale à 1 et toutes les autres égales à 0), le coefficient de partition est maximum et vaut 1.

1.7.5 Adaptation au clustering avec recouvrements

Il n'existe pas de mesure destinée à évaluer des pseudo-partitions. Cependant certains indices, parmi ceux cités précédemment, peuvent également être utilisés dans ce contexte. Par exemple :

Évaluation externe : les statistiques de *Rand*, *Jaccard* ou *Fowlkes et Mallows* s'adaptent plutôt bien à l'évaluation de partitions avec recouvrements : pour une paire d'objets (x_i, x_j), on observe si ces deux objets appartiennent à une même classe dans la partition de référence $\mathcal{P}$ et dans la partition obtenue $\mathcal{C}$. L'indice Γ (externe) permet également de considérer ce cas, bien que la valeur de l'indice soit diminuée par rapport à un schéma strict, puisqu'il y aura davantage de couples (x_i, x_j) appartenant à un même cluster. Enfin, la mesure d'entropie doit subir une légère modification : le dénominateur (n), pour jouer son rôle de normalisation, doit être changé en $\sum_{i=1}^{k_\mathcal{C}} |C_i|$.

Évaluation interne : Pour l'évaluation de partitions floues, l'indice Γ (interne) défini précédemment reste valide. De même, dans le cas de pseudo-hiérarchies, le coefficient de corrélation cophénétique se présente comme un bon indice de comparaison

entre la matrice cophénétique induite d'une pyramide et la matrice de proximité initiale. Notons au passage, que la généralisation d'une hiérarchie à une pyramide doit, mathématiquement, induire une meilleure correspondance entre ces deux matrices.

Évaluation relative : Le passage d'un schéma strict à un schéma avec recouvrements implique presque systématiquement une perte de qualité en ce qui concerne la dispersion intra et inter-clusters. En effet, les clusters sont élargis et s'intersectent, ils sont donc moins homogènes et moins séparés. L'indice de *Dunn*, par exemple, renverra systématiquement une valeur égale à 0 puisque la plus petite distance séparant deux clusters (numérateur) est nulle pour deux clusters qui s'intersectent. En revanche les indices de *Davies-Bouldin* et Γ (relatif) peuvent aider à comparer plusieurs schémas avec recouvrements. Le coefficient de partition apporte également une information concernant l'importance des recouvrements ; en considérant que chaque objet d'un cluster a une valeur d'appartenance ($u_k(x_i)$) égale à 1, ce coefficient prendra ses valeurs entre 1 (partition stricte) et t (correspondant à t clusters contenant chacun tous les objets).

Comme nous l'avons constaté, il existe de nombreuses mesures permettant d'évaluer, pour chaque type de schéma (strict, avec recouvrements ou flou) la pertinence des algorithmes de clustering. Cette évaluation peut être effectuée relativement à une connaissance *a priori* sur les données, en comparaison avec une matrice de proximité associée aux données ou "simplement" en mesurant les dispersions inter et intra-clusters. Cependant, ces indices ne permettent pas de comparer des schémas de natures différentes. Par exemple, un schéma avec recouvrement sera systématiquement moins pertinent qu'un schéma strict, relativement à l'homogénéité et à la séparation des clusters. Dans la suite, afin de motiver notre étude concernant le clustering avec recouvrements, nous serons amenés à envisager une autre stratégie d'évaluation. Pour cela nous utiliserons le regroupement de données à l'intérieur d'un processus plus général tel que l'apprentissage de règles de classification ou encore la classification de documents, afin d'observer l'influence des différents algorithmes et des différents types de schémas de clustering sur l'application globale.

1.8 Conclusion sur le clustering

Dans ce premier chapitre, nous avons présenté la problématique générale du clustering en définissant les trois étapes majeures de ce processus, à savoir : la préparation des données, le clustering proprement dit et l'exploitation des résultats de l'algorithme. Pour chacune de ces étapes, nous avons focalisé notre attention sur les grandes difficultés rencontrées, donnant lieu à des sous-domaines de recherche, comme par exemple : la pondération des variables descriptives, le choix d'une mesure de (dis)similarité ou d'une méthode de regroupement adaptée ou encore l'évaluation et la validation des schémas de clustering obtenus.

Nous avons surtout distingué les différents types de schémas envisageables suivant les méthodes existantes, et observé que la construction de hiérarchies ou partitions strictes est un résultat commun à la plupart des algorithmes. Les algorithmes de regroupement flou, proposant un résultat plus riche du point de vue informationnel, sont également très étudiés et peuvent facilement s'adapter si besoin, pour former des schémas stricts ou avec recouvrements (recours à des seuils). Ce dernier type de schémas (avec recouvrement) est peu étudié, sans doute parce que les approches envisagées jusqu'ici sont en majorité basées sur la recherche de clusters compacts et que les critères de qualité qui en découlent favo-

risent les schémas sans recouvrements. L'organisation de données en classes recouvrantes permet pourtant d'aboutir à des classes davantage représentatives de l'organisation réelle des données.

Dans le chapitre suivant, nous proposons une nouvelle approche de regroupement, dédiée à la construction d'une pseudo-partition. L'algorithme PoBOC répond à un cahier des charges contraignant puisqu'il tient compte de problématiques majeures du domaine telles que la faible sensibilité à la présence d'outliers, la construction de clusters de densités variées, la recherche du nombre de classes approprié et surtout l'existence d'intersections entre ces classes.

2

L'algorithme de clustering PoBOC

Sommaire

2.1 Motivations et cadre général de l'algorithme 54
2.2 Présentation de l'algorithme PoBOC 54
2.2.1 Présentation générale de PoBOC 54
2.2.2 Présentation formelle de l'algorithme PoBOC 56
2.3 Discussion sur l'algorithme PoBOC 65
2.3.1 Rappel du processus global 65
2.3.2 Positionnement de l'algorithme PoBOC 66
2.3.3 Traitement de grandes bases 68
2.4 Premières expérimentations 69
2.4.1 Analyse de PoBOC sur la base de données Iris 69
2.4.2 Évaluation de PoBOC sur d'autres bases de données 75
2.5 Conclusion 78

2.1 Motivations et cadre général de l'algorithme

La conclusion de la synthèse précédente concernant le clustering et les différentes approches proposées dans la littérature, est qu'il n'existe pas, ou peu, d'algorithmes adaptés à la construction de classes non-disjointes. Les formalisations mathématiques de la problématique du clustering, telles que les fonctions de coût à optimiser (*e.g.* méthodes de réallocations du type k-moyennes) ou la modélisation par mélanges de lois de probabilités, ne permettent pas de considérer des intersections entre classes, sans utilisation de seuils, plus ou moins arbitraires.

Du côté des méthodes hiérarchiques, l'approche pyramidale se pose comme une exception à la précédente conclusion, puisque le problème présente une formalisation mathématique. Cependant, la construction d'une pyramide est contrainte à la définition d'un ordre sur l'ensemble des objets ; chaque cluster final est alors défini comme un intervalle et s'intersecte avec au plus deux autres clusters (intervalles gauche et droit). Cette contrainte, indispensable à la visualisation d'une pyramide, limite considérablement les possibilités de schémas de clustering.

Certaines adaptations peuvent être envisagées pour "assouplir" un schéma de partitionnement stricte ou au contraire pour "restreindre" un schéma flou. Dans le premier cas, on peut proposer d'étendre les hyper-sphères obtenues par une méthode du type k-moyennes, en augmentant leur rayon. Dans la seconde alternative, il est possible de transformer les valeurs d'appartenance en affectations éventuellement multiples. Dans les deux cas, l'utilisation de seuils semble indispensable et le choix ou l'apprentissage de ces seuils devient une nouvelle problématique. De plus, comme nous le constaterons dans les expérimentations à venir, ces stratégies d'adaptation induisent souvent des intersections trop importantes.

Il convient alors de proposer une approche nouvelle, dédiée à la tâche de construction d'une pseudo-partition. Cette approche doit être générale, c'est à dire qu'elle ne doit pas être limitée à un ensemble bien précis d'applications, mais doit au contraire pouvoir traiter tout type de données dans des contextes applicatifs très différents. Nous proposons alors l'algorithme PoBOC (*Pole-Based Overlapping Clustering*), une méthode de clustering avec recouvrements, basée sur la construction d'un ensemble de pôles.

Le plan, pour la suite de ce chapitre, est le suivant : nous présentons en détail l'algorithme de clustering PoBOC en section 2.2. Nous proposons ensuite une discussion sur l'algorithme (section 2.3) et une présentation de quelques expérimentations préliminaires (section 2.4).

2.2 Présentation de l'algorithme PoBOC

Dans cette section, nous présentons l'algorithme PoBOC d'abord d'une manière générale, en donnant les grandes lignes de la méthode, puis formellement, en précisant chaque étape du processus.

2.2.1 Présentation générale de PoBOC

L'algorithme PoBOC se divise en trois étapes majeures :
- la construction d'un graphe de dissimilarité,
- la construction des pôles,
- la "multi-affectation" des objets aux pôles.

La construction d'un **graphe de dissimilarité** permet de représenter de manière condensée les objets et les relations entre les objets. Cette étape consiste alors à générer un graphe valué non-orienté tel que les sommets de ce graphe correspondent aux objets à traiter, et l'existence d'une arête entre deux objets indique que les deux objets sont "plutôt similaires"[1]. Chaque arête entre deux sommets est valuée par la valeur de la dissimilarité entre ces deux objets. Nous préciserons par la suite de manière formelle ce qui caractérise deux objets "plutôt similaires".

La construction des **pôles** est effectuée à partir du graphe de dissimilarité précédent et consiste à en extraire plusieurs sous-graphes (complets) qui constituent ce que l'on appellera alors par la suite les "pôles". La notion de pôle correspond à l'idée intuitive que l'on peut s'en faire : les pôles sont des ensembles d'objets (constitués d'au minimum un objet) distants les uns des autres et les objets constituant les pôles sont tous, deux à deux assez similaires, relativement à la densité locale. Une nouvelle fois, pour faciliter la compréhension de cette notion, nous proposons un exemple dans lequel les données sont représentées dans un espace de dimension 2. Dans la Figure 2.1, les trois ensembles d'objets P_1, P_2 et P_3 correspondent effectivement à la définition intuitive d'un pôle. Notons que les caractéristiques d'un outlier satisfont à cette définition (P_1).

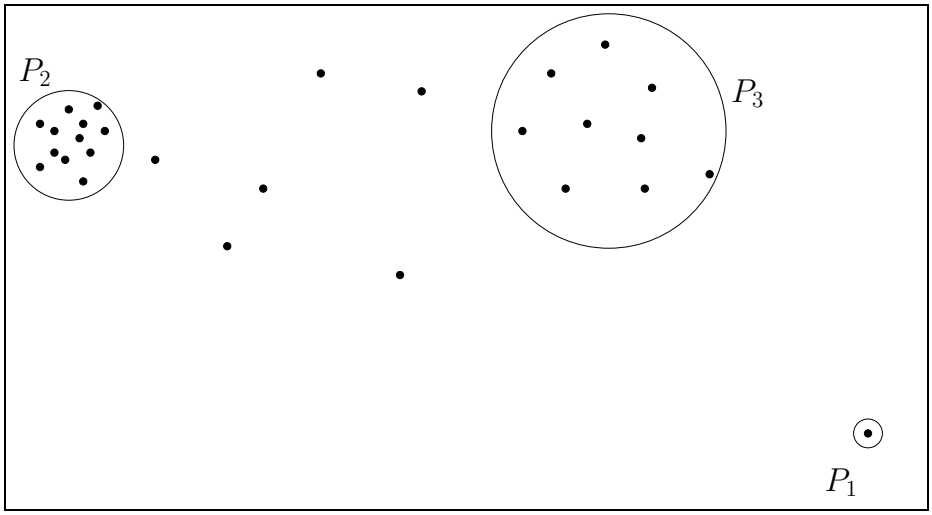


FIG. 2.1 – Exemples de pôles.

Enfin, l'étape de **multi-affectations** permet d'aboutir au schéma de clustering final par affectation des objets à un ou plusieurs pôles. C'est la possibilité d'affecter un même objet à plusieurs pôles qui permet d'obtenir des recouvrements entre clusters. La méthode utilisée pour y parvenir consiste à affecter d'abord l'objet au pôle qui lui est le plus proche, puis à envisager l'affectation au deuxième plus proche pôle et ainsi de suite jusqu'à ce qu'une affectation n'ait pas lieu. Un objet peut alors être affecté :

- soit uniquement à son plus proche pôle (affectation stricte),
- soit à ses pôles les plus proches (multi-affectations),
- soit à tous les pôles.

[1] Cette propriété distingue le graphe tel que nous l'envisageons des familles traditionnelles de "graphes de proximités" que nous présentons par la suite.

Le schéma final peut être une partition stricte dans le cas où chaque objet n'a été affecté qu'une seule fois.

2.2.2 Présentation formelle de l'algorithme PoBOC

La Figure 2.2 présente de façon formelle les différentes étapes de l'algorithme PoBOC. Dans cet algorithme, la notation $d_{moy}(x, X)$ correspond à la dissimilarité moyenne de l'objet x avec tous les autres objets de X (où $|X| = n$) :

$$d_{moy}(x, X) = \frac{1}{(n-1)} \sum_{x_i \in X \ , \ x_i \neq x} d(x, x_i) \tag{2.1}$$

De même la notation $d_{moy}(X, X)$ correspond à la dissimilarité moyenne entre deux objets de X :

$$d_{moy}(X, X) = \frac{2}{n.(n-1)} \sum_{i=2}^{n} \sum_{j<i} d(x_i, x_j) \tag{2.2}$$

On décrit maintenant les trois principales phases : construction du graphe de dissimilarité (étape 2), construction des pôles (étapes 3 à 7) et multi-affectations (étape 11).

Définition du graphe de dissimilarité

Le graphe que nous cherchons à construire s'apparente à la notion de graphes de proximités, présentée dans le chapitre 1 (section 1.6.7). En effet, dans les graphes RNG, GG ou encore DT, une arête entre deux objets indique une forte proximité, caractérisée par l'absence d'objets dans la "lune d'influence" des deux objets[2] (RNG), dans le "cercle d'influence" des deux objets (GG), ou l'adjacence des cellules de Voronoï associées (DT).

Cependant, la réciproque est fausse puisque l'on ne peut considérer dans ces familles de graphes, que l'absence d'arête entre deux objets signifie que ces deux objets sont "éloignés".

Nous choisissons d'alléger les conditions requises pour considérer que deux objets sont "proches", en introduisant la notion de voisinage d'un objet. Dans le graphe de dissimilarité (ou graphe de proximités) que nous définissons par la suite, la présence d'une arête entre deux objets x_i et x_j indique que x_i et x_j sont, en moyenne, plus similaires entre eux qu'avec les autres objets de X.

Réciproquement, l'absence d'arête entre deux objets x_i et x_j indique qu'au moins l'un de ces deux objets x_i ou x_j est, en moyenne, plus similaire aux autres objets de X.

Définition 2.1. *On définit le graphe de dissimilarité $G(X, V, d)$ par :*
- *l'ensemble de sommets X,*
- *l'ensemble des arêtes $(x_i, x_j) \in V$ telles que x_i et x_j sont "mutuellement voisins",*
- *la fonction de valuation d telle que*

$$d : V \to \mathbb{R}$$
$$(x_i, x_j) \to d(x_i, x_j) \text{ (mesure de dissimilarité)}$$

Nous formalisons cette notion de "mutuellement voisins" par l'équation 2.3 (Figure 2.2) qui oblige chacun des deux objets à appartenir au voisinage (strict) de l'autre. Si on note $\mathcal{N}(x)$ le voisinage d'un objet x, on a alors que :

[2] La "lune d'influence" de deux objets x_i et x_j correspond à l'intersection des deux hyper-sphères centrées respectivement en x_i et x_j et de rayon $\overline{x_i x_j}$.

Algorithme PoBOC : *Pole-Based Overlapping Clustering*

Entrée : Une matrice de dissimilarité D sur l'ensemble X des objets à traiter
Sortie : Une pseudo-partition $\mathcal{C} = \{C_1, \ldots, C_t\}$ de X

1. Initialisation : $\mathcal{P} = \emptyset$ (ensemble des pôles), $O = \emptyset$ (ensemble des objets couverts)

2. Construire le graphe de dissimilarité $G(X, V, d)$ ayant pour ensemble de sommets X et tel que

$$(x_i, x_j) \in V \text{ } ssi \text{ } d(x_i, x_j) < \min\{d_{moy}(x_i, X), d_{moy}(x_j, X)\} \tag{2.3}$$

3. Choisir un premier objet $\hat{x}_1 = \arg\max_{x_i \in X} d_{moy}(x_i, X)$, $k = 1$

4. Construction du $k^{ième}$ pôle P_k par `construction-pole`($\hat{x}_k$,$G(X, V, d)$)

5. Mise à jour : $\mathcal{P} = \mathcal{P} \cup \{P_k\}$ et $O = O \cup P_k$

6. Choisir un nouvel objet $\hat{x}_k = \arg\max_{x_i \in X \setminus O} d_{moy}(x_i, O)$

7. Si $d_{moy}(\hat{x}_k, O) > d_{moy}(X, X)$ alors $k = k + 1$ et retourner à l'étape 4

8. Restreindre les pôles à leur partie propre par suppression des objets partagés

9. Supprimer les éventuels pôles vides pour obtenir un ensemble $\tilde{\mathcal{P}} = \{\tilde{P}_1, \ldots, \tilde{P}_t\}$ de t pôles disjoints ($t \leq k$)

10. Initialiser chaque cluster $\{C_i = \tilde{P}_i\}_{i=1\ldots t}$

11. Pour chaque objet $x_i \in X \setminus O$: `affecter`(x_i,$\tilde{\mathcal{P}}$,D,$\mathcal{C}$)

12. Retourner $\mathcal{C}$ une pseudo-partition de X

FIG. 2.2 – L'algorithme de clustering PoBOC.

$$(x_i, x_j) \in V \; ssi \begin{cases} x_i \in \mathcal{N}(x_j) \\ \quad et \\ x_j \in \mathcal{N}(x_i) \end{cases} \tag{2.4}$$

Nous choisissons de délimiter le voisinage (strict) d'un objet x_i par la dissimilarité moyenne entre cet objet et les autres objets de X (rayon de voisinage de x_i : $d_{moy}(x_i, X)$). Ainsi, on dira que x_j est dans le voisinage de x_i si et seulement si x_i est en moyenne plus similaire à x_j qu'aux autres objets de X. Ce voisinage est défini par l'équation 2.5 :

$$x_j \in \mathcal{N}(x_i) \Leftrightarrow d(x_i, x_j) < d_{moy}(x_i, X) \tag{2.5}$$

La Figure 2.3 permet d'illustrer la contrainte d'existence d'une arête entre deux sommets. Les deux cercles délimitent les voisinages des deux objets ; la zone hachurée doit contenir les deux objets pour qu'ils soient reliés.

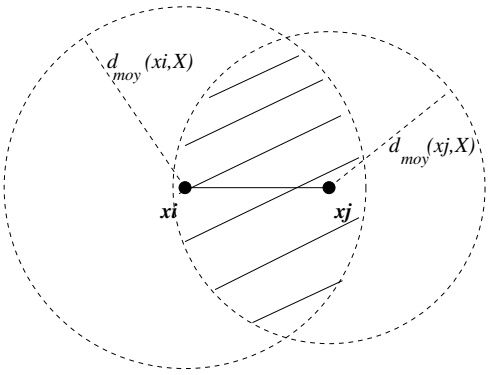


FIG. 2.3 – Contrainte sur l'existence d'une arête entre deux sommets dans $G(X, V, d)$.

Enfin, chaque arête (x_i, x_j) est valuée par la dissimilarité $d(x_i, x_j)$. Nous proposons, dans la suite, un exemple de construction d'un graphe de dissimilarité.

Exemple 2.1. On considère un ensemble d'objets $X = \{x_1, \ldots, x_{13}\}$ défini dans l'espace métrique $(\mathbb{R}^2, L_2)$, où L_2 désigne la distance Euclidienne. Ces points sont organisés conformément à la Figure 2.4. On observe sur cette figure que l'objet x_{10} est situé en marge des autres points, il peut être considéré comme un outlier. Les 12 autres points sont organisés en trois blocs de tailles et de densités différentes. On peut considérer les groupes suivants :

- $A = \{x_1, \ldots, x_5\}$ groupe de grande taille[3] et de densité forte,
- $B = \{x_6, \ldots, x_9\}$ groupe de taille moyenne et de densité faible,
- $C = \{x_{11}, \ldots, x_{13}\}$ groupe de petite taille et de forte densité.

On observe également que deux objets (x_6 et x_9) permettent de "joindre" respectivement les blocs A, B et B, C, alors que les blocs A et C sont clairement séparés.

La Figure 2.5 présente le graphe de dissimilarité obtenu sur ce même exemple[4].

L'objet x_{10} est isolé par la méthode de construction du graphe. En effet, la contrainte exprimée en 2.3 ne permet pas l'existence d'une arête entre x_{10} et les autres objets de X qui sont, en moyenne, plus proches entre eux que de x_{10}.

[3] On peut considérer ce groupe de taille importante puisqu'il contient plus d'un tiers des objets de X.

[4] Pour une plus grande clarté, les valuations des arcs ne sont pas mentionnées sur ce schéma.

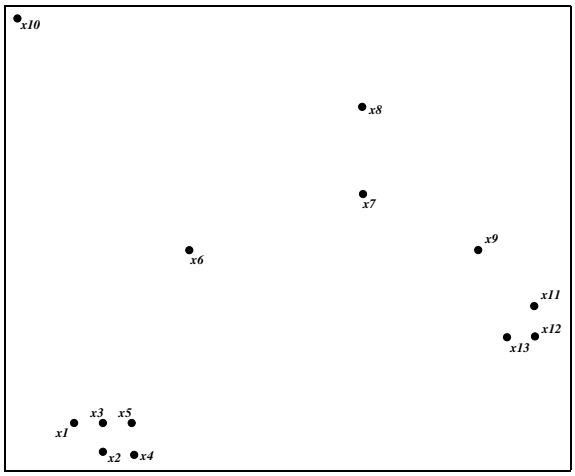


FIG. 2.4 – Organisation des objets de X dans $(\mathbb{R}^2, L_2)$.

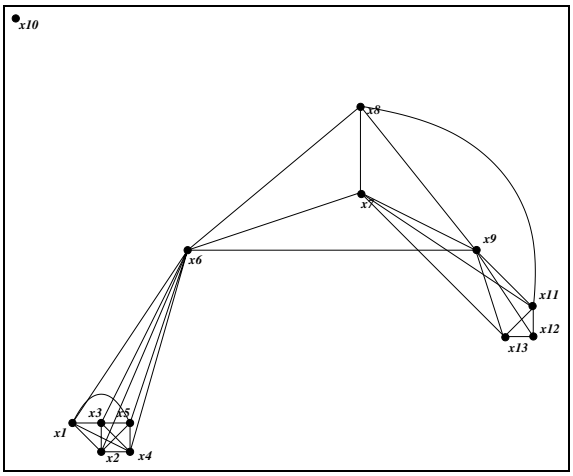


FIG. 2.5 – Graphe de dissimilarité de X.

Propriété 2.1. En généralisant, on montre que le graphe de dissimilarité, ainsi construit, permet d'isoler systématiquement une certaine catégorie d'outliers.

Nous rappelons ici la définition de Hawkins d'un $DB(m, \delta)$-*outliers*.

Définition 2.2. *Un objet $x^* \in X$ est un $DB(m, \delta)$-outlier s'il existe un sous-ensemble X' de X, constitué d'au moins m objets $x'_1, \ldots, x'_m$, tel que $\forall x'_i \in X',\ d(x^*, x'_i) > \delta$, où d est une mesure de dissimilarité définie dans l'espace de description $\mathcal{V}$.*

Ces outliers isolés (notés x^*) sont tels que m=*n-1* et $D = \max_{x_i \neq x^*} d_{moy}(x_i, X)$, où n désigne le nombre d'objets dans X.

Preuve

Soient $X = \{x_1, \ldots, x_n\}$ un ensemble d'objets, d une mesure de dissimilarité sur X, x^* un objet de X et $D = \max_{x_i \neq x^*} d_{moy}(x_i, X))$,

x^* est un $DB(n-1, D)$-*outliers*
$\Leftrightarrow \exists X' = \{x'_1, \ldots, x'_{n-1}\} \subset X$ t.q. $\forall x'_i \in X',\ d(x'_i, x^*) \geq D$, (par la définition 2.2)
$\Leftrightarrow \forall x_i \neq x^*,\ d(x_i, x^*) \geq D$, ($d(x^*, x^*) < D$ donc $x^* \notin X'$ d'où $X' = X \setminus \{x^*\}$)
$\Leftrightarrow \forall x_i \neq x^*,\ d(x_i, x^*) \geq \max_{x_j \neq x^*} d_{moy}(x_j, X)) \geq d_{moy}(x_i, X)$,
$\Rightarrow \forall x_i \neq x^*,\ d(x_i, x^*) \geq d_{moy}(x_i, X)$,
ce qui ne vérifie pas la contrainte d'existence d'une arête entre x_i et x^* dans le graphe de dissimilarité $G(X, V, d)$.

■

Si on s'intéresse, dans le graphe de la figure 2.5, à la composante connexe[5] $X \setminus \{x_{10}\}$, on observe plusieurs sous-graphes complets (ensemble de sommets tous deux à deux reliés par une arête) tels que $\{x_1, \ldots, x_6\}$, $\{x_6, \ldots, x_9\}$ ou encore $\{x_9, x_{11}, \ldots, x_{13}\}$. Ces trois sous-graphes complets sont révélateurs de l'organisation en trois blocs (A,B et C) mentionnée

[5]Dans un graphe G, une composante connexe C est définie par un ensemble de sommets $x_1, \ldots, x_m$, tel que toute paire de sommets (x_i, x_j) dans C est reliée par un chemin dans C.

précédemment. De plus, on observe davantage d'arêtes entre les sous-graphes correspondant aux blocs B et C ou A et B qu'entre A et C , confirmant ainsi l'observation faite sur l'organisation des blocs entre eux.

Enfin, on remarque que les sommets x_{12} et x_8 ne sont pas reliés par une arête alors que la dissimilarité entre ces deux objets est semblable à la dissimilarité entre les deux objets x_6 et x_9, qui eux, sont reliés dans le graphe. Ceci s'explique par le fait que l'objet x_{12} se situe dans une zone dense (plusieurs objets sont assez similaires à x_{12}), sont voisinage est alors plus restreint que les voisinages des objets x_6, x_8 ou x_9. Cette observation illustre bien l'intérêt d'utiliser la notion de densité (ou voisinage) plutôt qu'un même seuil pour l'ensemble des objets de X.

L'exemple 2.1 nous a permis d'illustrer les caractéristiques du graphe de dissimilarité ainsi construit. Ce graphe se caractérise par :
- l'isolement d'une certaine catégorie d'outliers,
- la mise en évidence de blocs d'objets plutôt similaires,
- la prise en compte des densités dans la définition de ces blocs.

Notons que la famille de graphes que nous venons de définir (notée GD : Graphe de Dissimilarité) n'est pas comparable (au sens de l'inclusion) aux graphes de proximités définis dans le chapitre précédent. En effet, la présence possible d'un sommet de degré nul interdit la relation $MST \subseteq GD$. D'autre part, la présence de cliques de degrés supérieurs à 2 interdit la relation inverse $GD \subseteq MST$.

Ce graphe de dissimilarité représente un condensé de la structure de l'ensemble des objets. Il constitue la base de la suite du processus de clustering et notamment de la génération des pôles.

Heuristique de choix des objets de départ

Les étapes 3 et 6 de l'algorithme 2.2 permettent de choisir d'abord un premier objet $\hat{x}_1$ (étape 3) puis d'itérer la recherche de nouveaux objets $\hat{x}_2, \ldots, \hat{x}_t$ (étape 6) jusqu'à vérifier un critère d'arrêt. Ces objets constitueront les sommets de départ de la construction de chaque pôle. L'heuristique, que nous utilisons, doit permettre de : (1) dégager des objets susceptibles de conduire à des pôles conformes à ce que nous avons présenté en 2.2.1 et (2) décider du nombre d'objets nécessaire et suffisant pour coller au mieux avec l'organisation réelle des données. Le choix de chaque objet n'est pas effectué à partir du graphe de dissimilarité précédemment généré.

Le premier objet choisi est celui qui vérifie la propriété suivante :

$$\hat{x}_1 = \arg\max_{x_i \in X} d_{moy}(x_i, X)$$

Il s'agit donc de l'objet en moyenne le plus dissimilaire aux autres (en particulier il pourra s'agir d'un outlier). Notons que cette technique est celle utilisée dans l'algorithme divisif hiérarchique DIANA [97] (cf.1.4) afin d'identifier l'objet le plus atypique d'un ensemble d'objets pour ensuite séparer cet ensemble en deux clusters. Le cas échéant, cet objet atypique peut donc être un outlier.

Un premier pôle P_1 est généré à partir du sommet $\hat{x}_1$ correspondant dans le graphe de dissimilarité. Nous préciserons, par la suite, la méthode de construction d'un pôle à partir d'un sommet de départ. Les objets du pôle P_1 sont alors stockés dans l'ensemble O qui sera, par la suite, réactualisé après chaque pôle construit, afin de contenir l'ensemble des objets déjà couverts par au moins un pôle.

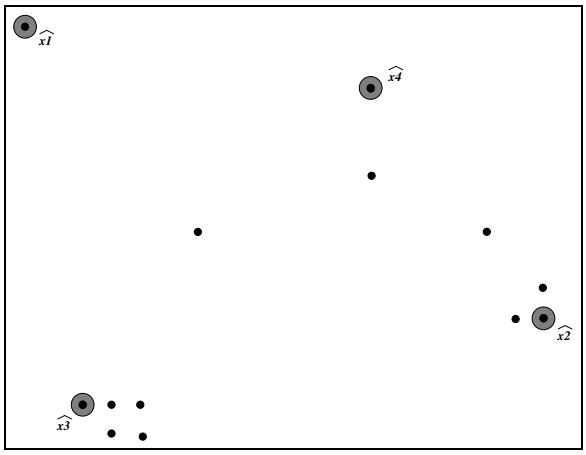


FIG. 2.6 – Heuristique de choix des objets de départ.

Un nouvel objet est ensuite choisi de façon à "s'éloigner" le plus possible du premier pôle P_1. En effet, P_1 constituera, par la suite, un pôle d'attraction où seront affectés des objets s'ils lui sont proches. On cherche alors à créer un second pôle d'attraction le plus distant possible du premier. Ce nouvel objet $\hat{x_2}$ et les objet sélectionnés qui suivront $\hat{x}_3, \ldots, \hat{x}_t$ seront alors les objets les plus dissimilaires à l'ensemble O :

$$\hat{x}_j = \underset{x_i \in X \setminus O}{\arg\max}\, d_{moy}(x_i, O)$$

Le processus de construction des pôles peut être stoppé par deux critères :
- soit l'ensemble O renferme tous les objets de X ($O = X$). Dans ce cas, tous les objets de X sont couverts par au moins un pôle ; il n'y a donc pas lieu de chercher une nouvelle zone inexplorée dans l'ensemble des données,
- soit le nouvel objet identifié n'est pas suffisamment distant des pôles déjà constitués. Formellement, un objet candidat $\hat{x}_j$ ne sera pas retenu si $d_{moy}(\hat{x}_j, O) \leq d_{moy}(X, X)$. Le processus sera stoppé puisqu'il n'existe pas d'autre objet $x \in X \setminus O$ tel que $d_{moy}(x, O) > d_{moy}(\hat{x}_j, O)$.

La figure 2.6 présente le nombre, l'ordre et le choix des objets sélectionnés comme sommets de départ pour la construction des pôles sur notre exemple. Le premier objet choisi correspond à l'outlier observé x_{10} puis les trois autres objets choisis appartiennent chacun à l'un des blocs dégagés (A, B et C). Les objets choisis et leur nombre correspondent donc à l'organisation observée préalablement. Nous verrons par la suite dans cet exemple, que le processus de sélection d'objets est stoppé par la première condition d'arrêt, autrement dit : $O = X$.

Heuristique de construction d'un pôle

Un pôle est constitué d'objets tous assez similaires les uns aux autres. Dans la représentation par graphe de dissimilarité, les sous-graphes complets (ou cliques) correspondent à cette notion. Nous allons alors chercher à extraire des cliques à partir du graphe de dissimilarité construit à l'étape 2. Par construction, dans ce graphe, une clique sera constituée de sommets tous en moyenne plus similaires entre eux qu'avec les autres sommets du graphe.

Propriété 2.2. *Soient $G(X, V, d)$ un graphe de dissimilarité sur X et $X' \subset X$ tel que le sous-graphe $G(X', V, d)$ est complet. Alors pour tout couple de sommets $(x'_i, x'_j) \in X' \times X'$ on a*

$$d(x'_i, x'_j) \leq \begin{cases} d_{moy}(x'_i, X) \\ \quad et \\ d_{moy}(x'_j, X) \end{cases}$$

La construction des pôles pourrait donc consister à rechercher les cliques de taille maximale dans le graphe de dissimilarité $G(X, V, d)$. Deux problèmes se poseraient alors :

- *Le nombre de pôles à construire :* il faudrait introduire un paramètre sur la taille minimale des cliques à extraire, sans pouvoir contrôler pour autant le nombre final de cliques ;
- *L'algorithme d'extraction :* il n'existe pas d'algorithme de recherche d'une clique de taille maximale dans un graphe, en temps polynomial. Ce problème est en effet reconnu comme étant NP-complet [17].

De plus, il y a un risque très élevé d'obtenir finalement des cliques quasi-identiques. La figure 2.7 illustre ce phénomène : l'absence d'une seule arête (entre les sommets x_1 et x_6) impliquerait l'extraction des deux cliques $\{x_1, \ldots, x_5\}$ et $\{x_2, \ldots, x_6\}$ qui sont de taille maximale dans le graphe.

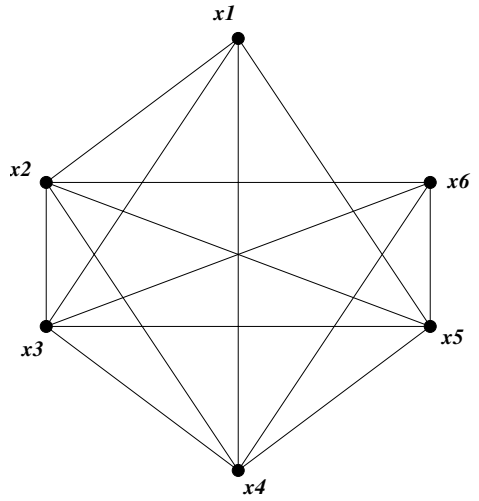


FIG. 2.7 – Exemple d'extraction de cliques quasi-identiques.

Compte-tenu de ces inconvénients, nous choisissons d'utiliser une heuristique de recherche permettant d'approximer la clique de taille maximale. Cette heuristique construit une clique à partir d'un sommet de départ. Nous utiliserons donc les objets $\hat{x}_1, \ldots, \hat{x}_t$ sélectionnés précédemment comme points de départ pour la construction de chaque clique.

Le fait d'imposer la construction des pôles via les objets $\{\hat{x}_i\}_{i=1\ldots t}$ permet de diminuer le risque de cliques quasi-identiques, vu en figure 2.7, et détermine le nombre de pôles qui seront générés. Par ailleurs, l'heuristique d'approximation de la clique maximale est séquentielle, gloutonne, et donc de complexité linéaire sur le nombre de sommets dans le graphe initial ; cette heuristique consiste en l'ajout successif du plus proche sommet contenu dans le voisinage du sous-graphe déjà construit, en partant du sous-graphe restreint à l'objet initial $\hat{x}_i$. Nous présentons plus en détail cette procédure en figure 2.8.

Procedure `construction-pole`$(\hat{x}, G(X,V,d))$

1. Initialisation : $P = \{\hat{x}\}$,
2. Construire le voisinage de P dans $G(X,V,d)$:

$$\mathcal{N}(P) = \{x \in X | \forall\ x_i \in P, (x, x_i) \in V\}$$

3. Tant que $\mathcal{N}(P) \neq \emptyset$ faire :

4. Ajouter à P l'objet x tel que $x \in \mathcal{N}(P)$ et $x = \underset{x_i \in \mathcal{N}(P)}{\arg\min}\, d_{moy}(x_i, P)$
5. Mettre à jour $\mathcal{N}(P)$

6. Retourner le pôle P.

FIG. 2.8 – La procédure de construction d'un pôle.

Sur notre exemple (graphe figure 2.5), nous rappelons que les objets sélectionnés sont : x_{10}, x_{12}, x_1 et x_8 :

- Partant de $P_1 = \{x_{10}\}$, le voisinage de P_1 est l'ensemble vide (car x_{10} est de degré nul dans $G(X,V,d)$). Il n'y a donc pas d'ajout possible, `construction-pole`$(x_{10}, G(X,V,d))$ retourne alors : $P_1 = \{x_{10}\}$;
- Partant de $P_2 = \{x_{12}\}$, le voisinage de P_2 est $\{x_9, x_{11}, x_{13}\}$ (tous reliés à x_{12}) ; le plus proche (x_{11}) est ajouté ($P_2 = \{x_{11}, x_{12}\}$) ; le voisinage est alors réduit à $\{x_9, x_{13}\}$ (tous deux reliés à x_{11} et x_{12}) ; le plus proche (x_{13}) est ajouté ($P_2 = \{x_{11}, x_{12}, x_{13}\}$) le voisinage est alors réduit à $\{x_9\}$ (relié à x_{11}, x_{12} et x_{13}) ; x_9 est finalement ajouté et le voisinage de P_2 devient vide. `construction-pole`$(x_{12}, G(X,V,d))$ retourne alors : $P_2 = \{x_9, x_{11}, x_{12}, x_{13}\}$;
- Partant de $P_3 = \{x_1\}$, de la même façon, `construction-pole`$(x_1, G(X,V,d))$ retourne : $P_3 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$;
- Partant de $P_4 = \{x_8\}$, `construction-pole`$(x_8, G(X,V,d))$ retourne : $P_4 = \{x_6, x_7, x_8, x_9\}$;

L'étape de restriction des pôles

Les pôles doivent être distants les uns des autres. Ce critère de "séparation" fait partie de la définition intuitive que nous avons donné initialement. En effet, plus les pôles seront séparés, moins il y aura d'ambiguïtés, par la suite, pour assigner les objets aux pôles. Or, par la méthode de construction des pôles, décrite en figure 2.8, ces derniers peuvent posséder certains éléments en commun. Le partage d'objets entre les pôles va à l'encontre du critère de séparation développé précédemment, nous choisissons alors de rectifier ces pôles en supprimant les objets partagés. Ces objets seront, ensuite, assujettis à l'étape de multi-affectation. On note $\tilde{P}_i$ la restriction du pôle P_i à ses objets caractéristiques.

La figure 2.9 présente la disposition des pôles $\tilde{P}_1, \ldots, \tilde{P}_4$ issus de la restriction de $P_1, \ldots, P_4$. Une nouvelle fois, on s'aperçoit sur cet exemple que les 4 pôles obtenus correspondent effectivement aux objets extrêmes de chacun des blocs observés initialement.

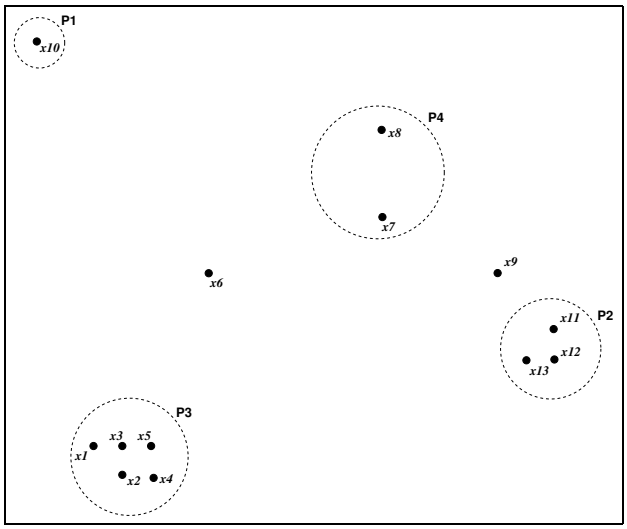


FIG. 2.9 – Pôles obtenus sur X.

Multi-affectations des objets aux pôles

L'étape de multi-affectations est cruciale dans le processus de clustering puisque c'est cette étape qui permet de construire les clusters qui seront retournés par l'algorithme. Il s'agit de parcourir l'ensemble des objets n'appartenant à aucun pôle, et pour chaque objet, de calculer sa valeur d'appartenance par rapport à chacun des pôles établis et de comparer ces valeurs pour choisir le ou les pôle(s) au(x)quel(s) l'objet doit être affecté.

Dans ce qui suit, on considère, pour simplifier, que la matrice de dissimilarité est normalisée dans l'intervalle $[0, 1]$. La valeur d'appartenance d'un objet x_i à un pôle $\tilde{P}_k$ est relative à la dissimilarité moyenne entre l'objet x_i et chacun des objets constitutifs de $\tilde{P}_k$. Cette valeur d'appartenance est alors définie par :

$$u(x_i, \tilde{P}_k) = 1 - d_{moy}(x_i, \tilde{P}_k)$$

Pour un objet x_i, la procédure d'affectation est décrite en figure 2.10. Elle consiste d'abord à ordonner les pôles par valeur d'appartenance décroissante, via la fonction bijective Φ. Ainsi $\Phi(1)$ correspond à l'indice du pôle le plus "proche" de x_i, $\Phi(2)$ à l'indice du deuxième pôles le plus proche et ainsi de suite. Une fois l'ordonnancement effectué, x_i est affecté systématiquement au pôle dont il est le plus proche. Pour que x_i soit affecté au $k^{ième}$ pôle le plus proche, il faut qu'il ait été affecté aux $k-1$ précédents, et que la valeur d'appartenance de x_i avec ce pôle $\tilde{P}_{\Phi(k)}$ soit supérieure à la valeur attendue, si l'on considère une décroissance linéaire entre les deux pôles immédiatement plus proche et moins proche.

Considérons par exemple l'affectation simple ou multiple de l'objet x_9 dans notre exemple. On calcule les valeurs d'appartenance de x_9 à chacun des pôles et on obtient : $u(x_9, \tilde{P}_1) = 0.2$, $u(x_9, \tilde{P}_2) = 0.81$, $u(x_9, \tilde{P}_3) = 0.38$ et $u(x_9, \tilde{P}_4) = 0.73$. L'ordonnancement des pôles produit alors la fonction Φ telle que $\Phi(1) = 2$, $\Phi(2) = 4$, $\Phi(3) = 3$ et $\Phi(4) = 1$.

- x_9 est affecté au cluster C_2 lié au pôle $\tilde{P}_2$.
- La figure 2.11 schématise les valeurs d'appartenance décroissantes de x_9. Sur ce schéma, on observe la décroissance linéaire autour du pôle $\tilde{P}_4$. Ceci se matérialise par le segment

Procedure `affecter`$(x_i, \tilde{\mathcal{P}}, D, \mathcal{C})$

1. Construire la fonction bijective d'ordonnancement $\Phi : \{1, \dots t\} \rightarrow \{1, \dots t\}$ telle que :
$$u(x_i, \tilde{P}_{\Phi(1)}) \geq u(x_i, \tilde{P}_{\Phi(2)}) \geq \cdots \geq u(x_i, \tilde{P}_{\Phi(t)})$$
2. Affecter x_i à $C_{\Phi(1)}$, $k = 1$

3. Tant que x_i a été affecté à $C_{\Phi(k)}$ et $k < t$ alors :
4. $\quad k = k + 1$,
5. $\quad$ si $k < t$ et $u(x_i, \tilde{P}_{\Phi(k)}) > \frac{1}{2}.(u(x_i, \tilde{P}_{\Phi(k-1)}) + u(x_i, \tilde{P}_{\Phi(k+1)}))$, affecter x_i à $C_{\Phi(k)}$,
6. $\quad$ si $k = t$ et $u(x_i, \tilde{P}_{\Phi(k)}) > \frac{1}{2}.u(x_i, \tilde{P}_{\Phi(k-1)})$, affecter x_i à $C_{\Phi(k)}$.

FIG. 2.10 – La procédure de multi-affectations d'un objet aux pôles.

en pointillés se trouvant sous la ligne brisée. On peut donc observer que la valeur d'appartenance de x_9 au pôle $\tilde{P}_4$ (0.73) est supérieure à la valeur que l'on obtiendrait si l'on considérait cette décroissance linéaire entre $\tilde{P}_2$ et $\tilde{P}_3$ (0.59). x_9 vérifie les conditions d'affectation au cluster C_4 correspondant au pôle $\tilde{P}_4$. Il est donc affecté à ce cluster.
- En revanche, ces conditions ne sont pas vérifiées pour le pôle $\tilde{P}_3$: la valeur d'appartenance $u(x_9, \tilde{P}_3) = 0.38$ est inférieure à la valeur que l'on obtiendrait en considérant une décroissance linéaire entre $\tilde{P}_4$ et $\tilde{P}_1$ (0.47). x_9 n'est donc pas affecté à ce cluster.
- Puisque x_9 n'a pas été affecté à l'étape précédente, il ne peut plus être affecté à d'autres clusters.

La figure 2.12 présente le schéma de clustering final obtenu par l'algorithme PoBOC sur notre exemple. On peut donc observer que les 4 clusters construits correspondent bien aux 4 blocs attendus avec la particularité que les objets x_6 et x_9 sont partagés par deux blocs "assez proches".

Cette méthode d'affectation permet d'éviter l'utilisation d'un même seuil d'affectation pour tous les couples : (objet, pôle).

2.3 Discussion sur l'algorithme PoBOC

2.3.1 Rappel du processus global

Dans le schéma de la figure 2.13, nous rappelons le processus global de clustering avec l'algorithme PoBOC. Nous retrouvons dans ce schéma les trois grandes étapes pré-citées de construction d'un graphe de dissimilarité, construction de pôles et multi-affectations des objets aux pôles. Observons alors que la matrice de dissimilarité entre objets constitue la seule donnée en entrée de l'algorithme (pas d'autres paramètres) qui produit alors un ensemble de clusters non-disjoints.

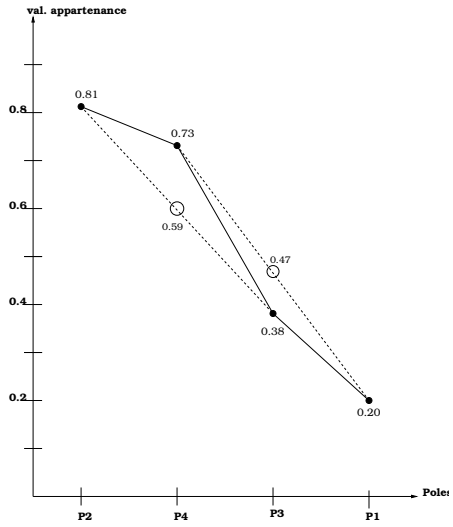


FIG. 2.11 – Exemple de multi-affectations.

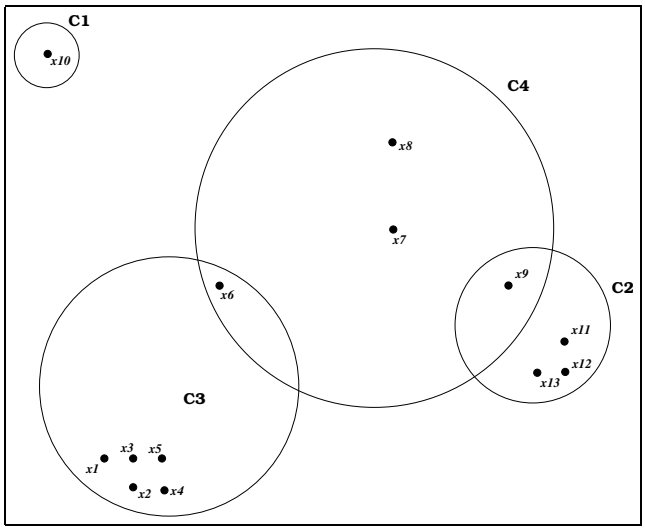


FIG. 2.12 – Schémas de clustering final.

2.3.2 Positionnement de l'algorithme PoBOC

L'algorithme PoBOC retourne une pseudo-partition de l'ensemble des objets de départ. A ce titre, on peut considérer cet algorithme de clustering comme un algorithme de partitionnement. De plus, une comparaison peut être faite entre PoBOC et l'algorithme des k-moyennes : ce dernier produit une partition par étapes successives de réallocations simples des objets à des foyers, tandis que PoBOC consiste en une étape unique d'affectations simples ou multiples des objets à des pôles. Les "foyers" utilisés dans l'algorithme des k-moyennes et ses variantes sont représentatifs des clusters. Il s'agit le plus souvent d'un objet (centroïde ou médoïde) central pour chaque cluster. Dans PoBOC, les pôles sont constitués d'un ou plusieurs objets "extrêmes" généralement éloignés des centres des clusters. Enfin, les itérations successives de réallocations, utilisées dans la méthode des k-moyennes, sont remplacées dans PoBOC par une unique étape précédée d'une initialisation non-aléatoire relativement précise. L'algorithme PoBOC ne repose donc pas sur une base formelle d'optimisation d'une fonction de coût. Ce type d'approches par optimisation nous semble difficilement utilisable dans le cas d'un algorithme dédié au clustering avec recouvrements.

Notons, de plus, que l'algorithme PoBOC utilise d'autres notions et formalismes inspirés des approches de clustering à partir de graphes et de clustering basés sur les densités. Le formalisme des graphes est utilisé pour extraire une représentation condensée de la matrice de dissimilarité et pour générer des sous-graphes particuliers conduisant aux pôles. Ce même graphe de dissimilarité est construit de façon à intégrer la notion de voisinage d'un point, notion empruntée aux méthodes utilisant les densités.

Outre le fait de proposer un schéma de clustering recouvrant, PoBOC présente les intérêts suivants :

- l'algorithme est peu sensible à la présence d'outliers,
- le nombre de clusters n'est pas fixé *a priori*,
- l'algorithme ne nécessite pas de paramètres externes ou de phases d'apprentissage de tels paramètres.

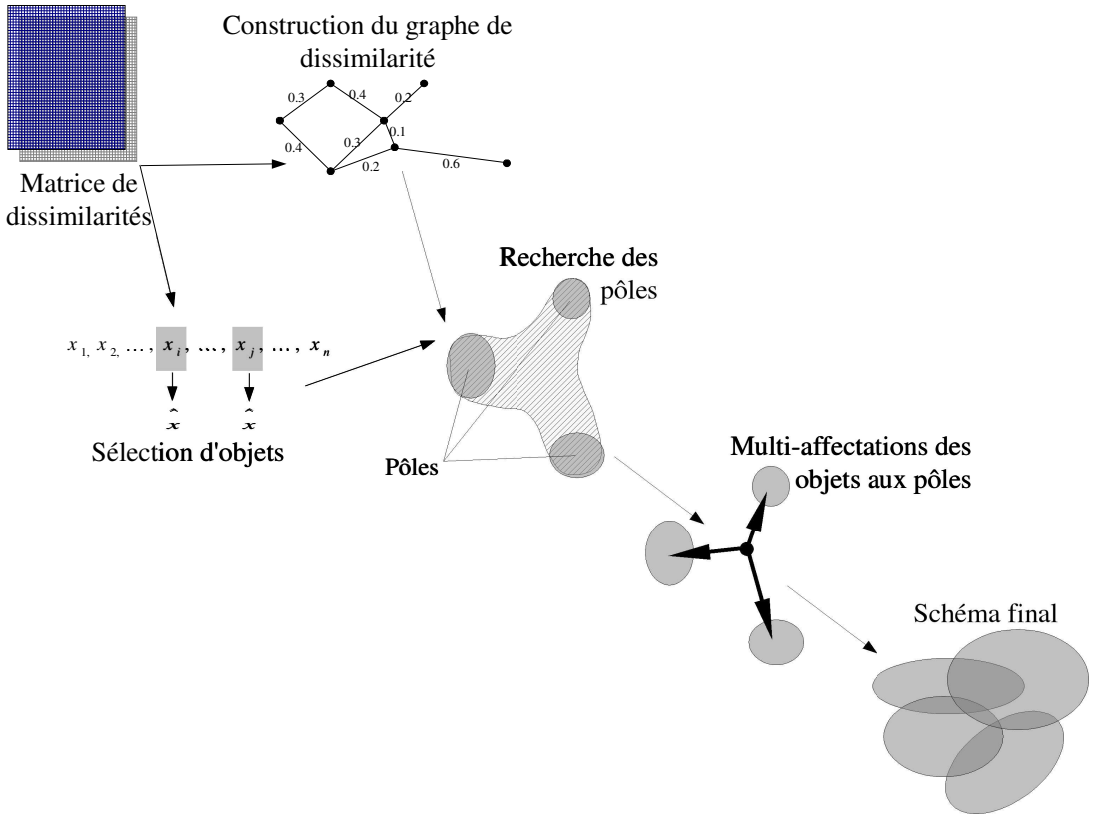


FIG. 2.13 – Processus global de PoBOC.

Nous avons vu que la méthode de construction du graphe de dissimilarité permet le plus souvent d'isoler un outlier. Cet isolement a pour effet de constituer un pôle restreint à cet objet encourageant ainsi la formation d'un cluster restreint, lui aussi, à cet outlier. Cependant, on ne peut pas affirmer que l'algorithme PoBOC soit totalement insensible aux outliers ; en effet, la présence d'un objet très différent du reste de la population peut entraîner une augmentation sensible de la dissimilarité moyenne du système ($d_{moy}(X, X)$). Ce paramètre interne, qui pourrait donner lieu à discussion[6], intervient activement dans la décision concernant le nombre de pôles générés. De façon indirecte, un tel outlier pourrait alors favoriser la construction d'un nombre de clusters inférieur à ce qui est attendu. Notons que plus un outlier affecte le résultat de PoBOC, plus il est facile à détecter ; il pourrait alors être supprimé par un pré-traitement simple des données, avant le processus de clustering.

Par la suite, nous discutons du coût de chacune des étapes de l'algorithme PoBOC, en terme de complexité. Cette analyse permet alors d'aborder le traitement de très grandes bases de données (VLDB[7]).

[6] Discussion notamment par rapport à l'avantage mentionné concernant l'absence de seuils externes.
[7] *Very Large DataBases*

2.3.3 Traitement de grandes bases

Nous analysons, ci-dessous, chacune des étapes de l'algorithme PoBOC afin de calculer la complexité globale de la méthode. Pour simplifier, nous proposons une complexité au pire cas, majorant donc la complexité réelle de la méthode. Ce faisant, nous n'utilisons que les paramètres n et k correspondant respectivement, à la taille de l'ensemble d'objets à traiter et au nombre de pôles générés par la méthode.

Construction du graphe de dissimilarité : (étape 2) Cette étape nécessite d'abord un pré-calcul des $\{d_{moy}(x_i, X)\}_{i=1...n}$. Ce calcul s'effectue en $O(n^2)$, tout comme la construction de l'ensemble V d'arêtes, nécessitant un parcours des $\frac{n.(n-1)}{2}$ paires d'objets. La construction du graphe est donc de complexité $O(n^2)$;

Construction des pôles : (itération des étapes 3 à 7) Chaque pôle est généré à partir d'un sommet. Ce sommet est sélectionné parmi les $n - |O|$ objets non-couverts, en calculant la dissimilarité moyenne $d_{moy}(x, O)$. La sélection du sommet nécessite alors $(n-|O|).|O|$ comparaisons ; la complexité de cette étape est donc majorée par $O(n^2)$. Ensuite, la construction d'une clique à partir de ce sommet est, au pire cas, également de complexité quadratique sur le nombre n d'objets. Ce pire cas correspond à un graphe de dissimilarité complet, ce qui ne correspond pas à une situation réelle. Ces deux étapes successives de choix d'un sommet de départ et de construction d'un pôle sont répétées k fois. La complexité de cette super-étape est donc majorée par $O(k.n^2)$.

Restriction des pôles : (étape 8) La restriction des pôles consiste à rechercher (parmi les k pôles) et à supprimer les objets (parmi n) qui apparaissent dans plusieurs pôles. Cette étape nécessite donc $k.n$ comparaisons (complexité en $O(k.n)$).

Multi-affectations des objets aux pôles : (étape 11) Cette dernière étape implique de considérer successivement les n objets, à calculer pour chaque objet ses k valeurs d'appartenance aux pôles, puis à comparer ces k valeurs pour l'affecter. Le calcul des valeurs d'appartenance pour un objet est de complexité linéaire sur le nombre d'objets, et l'étape d'affectation est de complexité linéaire sur le nombre de pôles. Ainsi, la complexité globale de cette étape est majorée par $O(n^2)$.

La complexité de l'algorithme PoBOC est donc majorée par $O(kn^2)$, correspondant à l'étape la plus coûteuse, à savoir la construction des pôles. Cet ordre de complexité relaie PoBOC au rang des approches plutôt proscrites dans le cas de larges ensembles de données, tout comme les algorithmes CLARANS et CURE ou encore les méthodes hiérarchiques du lien complet ou du lien moyen.

Cependant, même les méthodes de complexité linéaire, telles que l'algorithme des k-moyennes, sont jugées trop coûteuses dans le cas de très grands ensembles de données car elles nécessitent plusieurs parcours de la base. Ainsi [134] présentent différentes stratégies de modifications de l'algorithme, afin de traiter ces grands ensembles :

- proposer une approche incrémentale de l'algorithme,
- utiliser une stratégie du type "diviser pour régner",
- utiliser une représentation intermédiaire.

L'approche incrémentale considère chaque objet successivement en l'intégrant à un cluster existant ou en constituant un nouveau cluster. Inversement, les stratégies "diviser pour régner" considèrent l'ensemble des objets, divisés en plusieurs blocs, sur chacun desquels un processus de clustering est appliqué avant d'en fusionner les résultats. Enfin,

l'utilisation d'une représentation intermédiaire concerne la description initiale des données et dépend alors de l'application envisagée.

Le chapitre 5 présente l'utilisation de l'algorithme PoBOC dans le cadre du regroupement de mots, extraits de larges corpus de documents. On montrera, à cette occasion, que l'algorithme peut être modifié afin de traiter de façon incrémentale une grande quantité d'objets.

La complexité en espace mémoire est également un facteur à considérer. L'utilisation de PoBOC nécessite le stockage de la matrice de (dis)similarité. D'une manière générale, les algorithmes qui se basent sur une description relationnelle des données (matrice de (dis)similarité) présentent cet inconvénient, contrairement aux méthodes traitant des données dites "données objet". Nous verrons, toujours dans le chapitre 5, qu'une approche incrémentale de l'algorithme PoBOC permet de limiter l'espace de stockage nécessaire.

2.4 Premières expérimentations

2.4.1 Analyse de PoBOC sur la base de données Iris

Dans un premier temps, nous observons le comportement de l'algorithme PoBOC sur une base de données réelle, souvent utilisée en apprentissage. La base Iris fait partie des jeux de données de l'UCI repository [122], généralement utilisés pour l'apprentissage supervisé. Cette base de données contient 150 descriptions de fleurs appartenant à la famille des Iris, à l'aide des quatre attributs numériques suivants : longueur et largeur des sépales, longueur et largeur des pétales (en cm). Ces 150 iris sont équitablement répartis en trois classes : les iris *Setosa*, *Versicolor* et *Virginica*.

Nous avons choisi cette base pour trois raisons majeures :

- la faible dimension de l'espace de description et la quantité raisonnable de données permettent de les visualiser,
- les attributs, tous numériques, autorisent l'utilisation d'une mesure de distance classique (*e.g.* la distance euclidienne),
- l'organisation *a priori*, en trois classes, fournit une base d'évaluation complémentaire à la visualisation. Cette information n'est bien sûr pas utilisée lors du processus de clustering.

Dans la suite, nous utilisons la distance euclidienne comme mesure de comparaison entre les objets. La figure 2.14 présente le résultat de l'étape de construction des pôles par PoBOC. Appelons x, y, z et t les dimensions de l'espace des données. Les quatre graphiques correspondent chacun à la visualisation des données selon trois des quatre axes de l'espace, chaque pôle étant caractérisé par une même légende sur chacune des configurations. On note, tout d'abord, que PoBOC construit au total 3 pôles, conformément à la classification préétablie. Les pôles étant générés à partir d'un objet sélectionné ; les segments en pointillés nous aident à visualiser les 3 objets sélectionnés successivement. D'une manière générale, les graphiques (c) et (d) offrent les meilleurs angles de vue pour observer ces pôles.

Le premier pôle, matérialisé par les "+", se situe dans la partie supérieure droite des quatre graphiques. Le trait en pointillés indique le point de départ du second pôle (matérialisé par les "×"). On constate alors que le second pôle est construit à l'opposé du premier, et renferme un ensemble d'objets clairement séparés du reste, notamment sur les graphiques (c) et (d). Les étoiles indiquent la position du troisième et dernier pôle, formant un nuage encadré par les 2 premiers pôles.

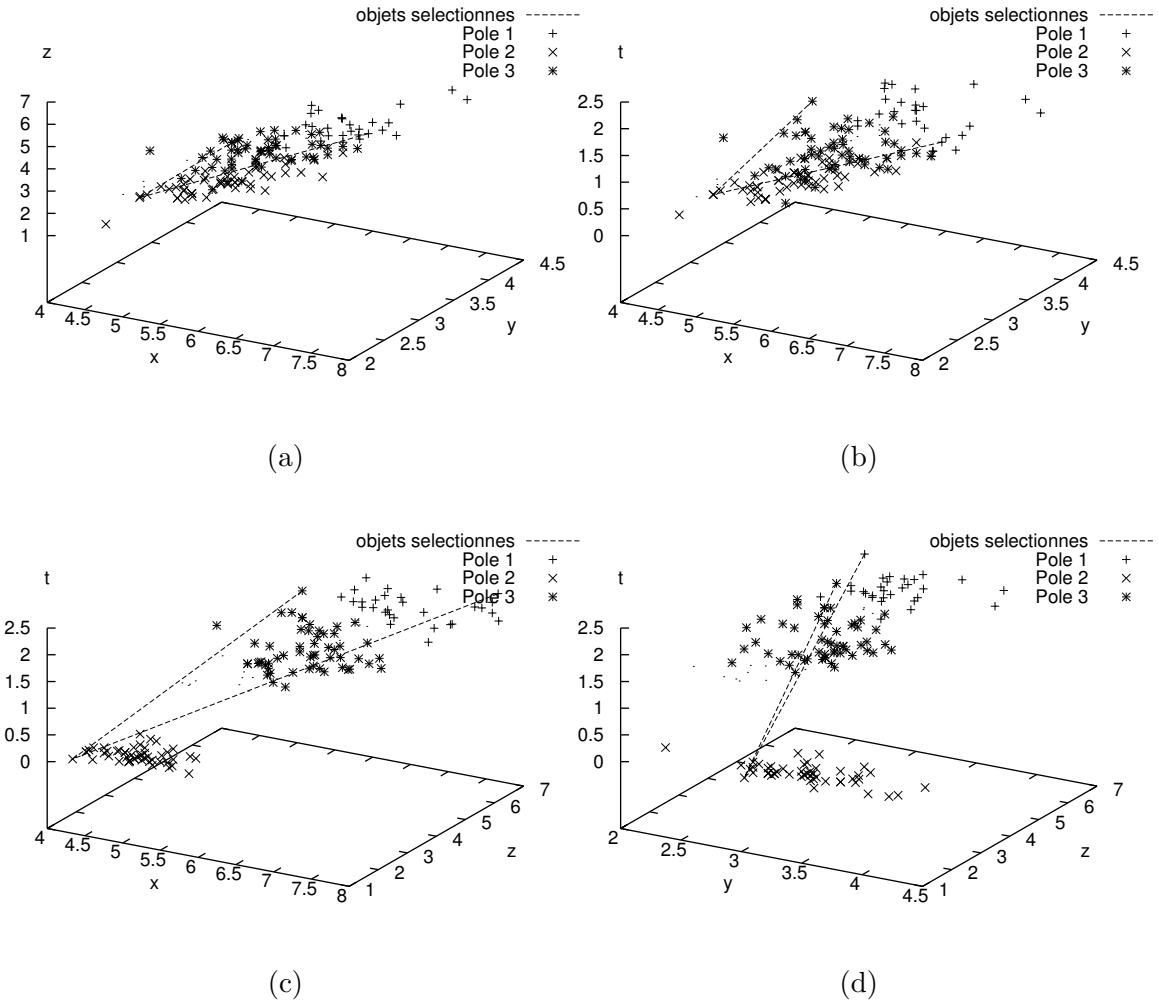


FIG. 2.14 – Observations des pôles constitués par PoBOC sur la base de données Iris selon les coordonnées : x, y, z (a), x, y, t (b), x, z, t (c) et y, z, t (d).

La figure (c), entre autre, permet de visualiser clairement les caractéristiques des pôles générés. D'abord, les trois pôles correspondent à des ensembles d'objets séparés, ce qui vérifie le premier critère de la définition intuitive de pôles. Enfin, on note que ces pôles n'ont pas tous la même densité ; en effet le deuxième pôle généré (désigné par les "×") est constitué d'objets plus rapprochés que les deux autres pôles. La densité des pôles reflète la densité locale de la région de l'espace dans laquelle ils sont situés. Cette observation est satisfaisante, par rapport au deuxième critère caractéristique des pôles.

Finalement, on note sur ces figures, que le premier et le troisième pôle sont assez proches. On peut alors s'attendre à ce que les clusters issus de ces deux pôles possèdent plusieurs objets en commun, et que le cluster issu du second pôle partage peu d'objets avec les autres.

Cluster n°	Taille	Nb objets exclusifs	intersections		
1	39	29	-	4	10
2	54	50	4	-	4
3	71	61	10	4	-

TAB. 2.1 – Analyse des clusters obtenus par PoBOC sur la base de données Iris.

Le tableau 2.1 introduit une première analyse des clusters obtenus par PoBOC. Chaque ligne correspond à un cluster et les informations sur sa taille, sur le nombre d'objets n'appartenant qu'à ce cluster et sur ses intersections avec les 2 autres clusters, sont répertoriées. Par exemple, le cluster n° 1 renferme 39 objets dont 29 sont exclusivement dans ce cluster. Il s'intersecte avec les clusters n° 2 et n° 3 et ces intersections sont respectivement de taille 4 et 10.

Par ce tableau, on confirme les hypothèses formulées précédemment : les clusters n° 1 et n° 3 possèdent d'avantage d'objets en commun (10) que les clusters n° 1 et n° 2 ou n° 2 et n° 3. Globalement, on observe que les clusters produits par PoBOC peuvent être de tailles variées (de 39 à 71 objets) avec plus ou moins d'intersections (de 5% à plus de 25% d'objets partagés).

On souhaite à présent évaluer la qualité du schéma obtenu par PoBOC sur la base Iris. Pour cela, nous allons analyser ce schéma d'abord en le comparant à la classification de référence puis en évaluant sa qualité intrinsèque. Pour ces deux stratégies d'évaluation nous observerons trois aspects du schéma issu de PoBOC : le nombre de clusters, la pertinence des clusters et la qualité des intersections entre clusters. L'algorithme des k-moyennes sera alors utilisé comme base de comparaison pour l'ensemble de l'étude.

Évaluation du schéma comparativement à la classification de référence

Clusters \ Classes	*setosa*	*versicolor*	*virginica*
n° 1	0	4	**35**
n° 2	**50**	4	0
n° 3	0	**50**	21

TAB. 2.2 – Table de contingence sur Iris.

La table de contingence (table 2.2) indique que chaque cluster représente une classe différente d'iris. La première classe d'iris est plutôt bien retrouvée puisque le cluster n ° 2 contient toutes les instances de cette classe plus quelques instances (4) de la seconde. Le cluster n ° 1 correspond à une sous-partie de la troisième classe d'iris et le cluster n ° 3 renferme la totalité des iris de la deuxième classe plus le complément de la troisième.

Outre le fait que PoBOC propose un schéma en trois clusters, conformément à la classification de référence, l'analyse de la table de contingence met en évidence que ces trois clusters correspondent assez bien aux trois classes attendues. Cette "correspondance" est alors quantifiée dans le tableau 2.3 par des indices d'évaluation externe.

Méthode	Stat. de *Huberts (externe)*↗	Statistique de *Rand* ↗	Stat. de *Huberts (interne)*↗
k-moyennes ($k = 3$)	0.146±0.006	0.842±0.074	0.339±0.021
PoBOC sans recouv^{ts}	0.153	0.874	0.350
PoBOC	0.147	0.839	0.343
Classification de réf.	0.160	1.000	0.313

TAB. 2.3 – Évaluation du schéma par des indices externes.

Dans ce tableau, trois schémas de clustering sont comparés (trois premières lignes) : l'algorithme des k-moyennes[8] pour un nombre de clusters identique à PoBOC, la version "stricte" de l'algorithme PoBOC[9] et l'algorithme PoBOC original. La version stricte de PoBOC est utile, d'une part pour comparer des schémas de même nature avec l'algorithme des k-moyennes (schémas strictes), et d'autre part pour observer l'influence des intersections entre clusters sur leur qualité.
Les deux indices externes utilisés sont :

- l'adaptation de la statistique de *Huberts* (Γ) pour comparer un schéma de clustering et une classification préétablie. Cet indice compare, pour chaque paire d'objets, la distance entre ces deux objets dans le schéma de clustering (distance entre les clusters respectifs) et dans la classification préétablie (distance entre les classes respectives). L'indice est alors maximum pour un schéma de clustering identique à la classification.
- Considérant que deux objets sont liés s'ils appartiennent à un même cluster, la statistique de *Rand* compte la proportion de liaisons ou non-liaisons "correctes" dans le schéma de clustering. Une liaison est "correcte" si les deux objets appartiennent à la même classe de référence. Une non-liaison est "correcte" si les deux objets (non-liés) appartiennent à des classes différentes. La valeur maximale 1.0, sur cet indice, indique que le schéma de clustering est identique à la classification.

Les mesures reportées dans le tableau indiquent que le schéma proposé par la version stricte de l'algorithme PoBOC est équivalent aux meilleures partitions obtenues par k-moyennes. Les intersections entre clusters induisent naturellement une perte de qualité des clusters. Cette perte reste cependant raisonnable puisque la pseudo-partition obtenue par PoBOC est comparable aux schémas stricts résultant des k-moyennes, sur les deux indices.

[8]Les résultats présentés correspondent à des moyennes sur 1,000 itérations de l'algorithme k-moyennes avec différentes initialisations.

[9]La version stricte diffère de l'algorithme original par une étape d'affectations simples (au plus proche pôle) au lieu de l'étape de multi-affectations.

Enfin, la dernière colonne du tableau 2.3 donne une indication sur la correspondance entre les schémas de clustering et la matrice des distances, *via* la version "interne" de Γ. On observe alors que les schémas les plus représentatifs de l'information contenue dans la matrice des distances, sont ceux obtenus par PoBOC. Enfin, la dernière ligne du tableau précise que la classification préétablie ne correspond pas nécessairement à la matrice des distances. Cette dernière observation nous incite à compléter l'évaluation des schémas de clustering indépendamment de la classification proposée.

Évaluation de la qualité intrinsèque du schéma

Indépendamment de toute connaissance extérieure, un bon schéma de clustering est caractérisé par des clusters compacts et bien séparés. Comme nous l'avons mentionné dans le chapitre précédent, ces deux caractéristiques sont à la base de la plupart des indices de qualité. Nous utilisons alors les mesures d'inertie intra-cluster et inter-clusters permettant de quantifier respectivement la dispersion des objets à l'intérieur des clusters (indice à minimiser $\searrow$) et la dispersion des clusters entre eux (indice à maximiser $\nearrow$). La statistique de *Huberts* (Γ), adaptée à une évaluation comparative de plusieurs schémas, permet de combiner ces deux indicateurs (indice à maximiser $\nearrow$).

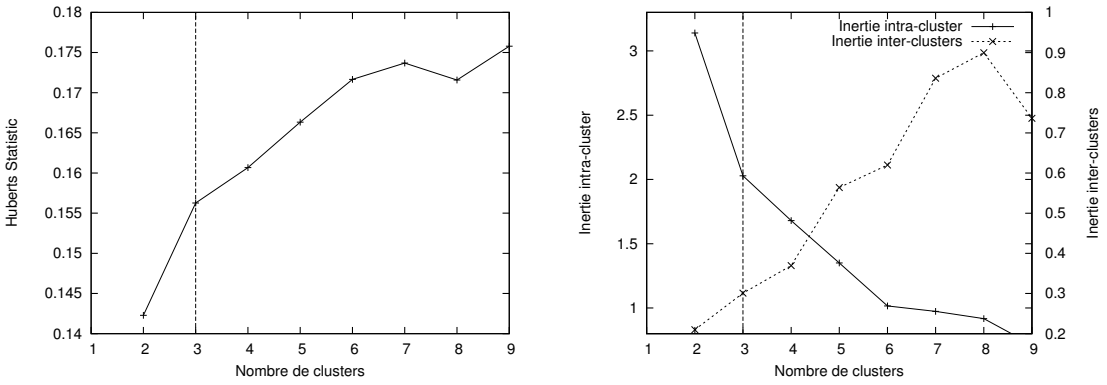


FIG. 2.15 – Évaluation du nombre de clusters sur la base Iris.

Ces trois indices sont utilisés pour détecter le/un nombre optimal de clusters organisant au mieux un ensemble de données. Sur la base Iris, l'algorithme des k-moyennes est exécuté plusieurs fois en faisant varier le paramètre k et donne lieu aux tracés des indices présentés en figure 2.15.

Les tracés des indices sont naturellement (quasiment) monotones. En effet, la qualité des schémas augmente avec le nombre de clusters autorisés. Sur ces tracés, les "pics" ou "cassures" observés, signifient que le paramétrage correspondant est particulièrement adapté ou au contraire inadapté, selon qu'ils correspondent à une amélioration ou à une détérioration de la qualité du schéma. Par exemple, le tracé de l'indice Γ sur Iris comporte deux "cassures" : à $k = 3$ (indiquant un meilleur schéma) et $k = 8$ (indiquant un moins bon schéma). Le tracé de l'intertie intra-cluster fait émerger les paramètres $k = 3$ et $k = 6$ tandis que le dernier indice distingue plutôt les schémas obtenus pour $k = 5$ et $k = 7$. La fusion de ces trois résultats élit le paramétrage $k = 3$ comme nombre optimal de clusters, ce qui valide à la fois le schéma obtenu par PoBOC et la classification de référence.

A titre d'information, notons que l'algorithme de clustering probabiliste AutoClass [21] qui recherche automatiquement ce nombre optimal de clusters propose, avec les meilleures

probabilités, les deux résultats suivants :

```
PROB exp(-2471.353) Nb clusters = 2
PROB exp(-2479.274) Nb clusters = 3
```

Les trois indices précédents sont de nouveaux employés pour comparer les schémas obtenus par les différentes méthodes de clustering :

Partitionnements stricts : Les schémas sans recouvrement, obtenus par k-moyennes et par la version stricte de PoBOC, sont comparés.

Partitionnements avec recouvrements : Une variante de k-moyennes proposant des clusters avec recouvrements est proposée comme élément de comparaison avec l'algorithme PoBOC. Cette variante est obtenue en remplaçant l'étape de réallocation originale de k-moyennes par une étape de réallocation multiple identique à la procédure de multi-affectations utilisée dans PoBOC.

Les résultats de cette étude sont rapportés dans le tableau 2.4 et complétés par le coefficient de partition qui mesure l'importance des recouvrements entre clusters : un coefficient égal à 1.0 correspond à un schéma strict tandis qu'une valeur égale à t (nombre de clusters) correspond à un taux de recouvrement maximum (tous les clusters sont alors identiques et renferment l'ensemble des objets).

Méthode	Inertie intra-cluster↘	Inertie inter-clusters↗	Stat. de *Huberts (relatif)*↗	Coefficient de partition↘
k-moyennes ($k = 3$)	2.05±0.56	0.291±0.073	0.155±0.007	1.00
PoBOC sans recouv^{ts}	1.68	0.279	0.163	1.00
PoBOC	3.06	0.257	0.157	1.09
k-moyennes +recouv^{ts}	7.32±4.26	0.261±0.147	0.106±0.058	1.66±0.28

TAB. 2.4 – Évaluation du schéma par des indices relatifs.

Les mesures confirment les conclusions précédentes, à savoir la présence de clusters plus compacts avec PoBOC (inertie intra-classe plus faible) et séparés de façon comparable (inertie inter-clusters), dans le cas des schémas stricts . Ces deux premières lignes du tableau montrent finalement que les clusters générés par PoBOC sont meilleurs que ceux générés par k-moyennes (Γ plus élevé).

Enfin, les deux lignes inférieures nous informent sur la quantité et la qualité des recouvrements produits par chacune des deux méthodes de clustering. La stratégie de construction des pôles, proposée dans PoBOC apparaît alors décisive puisque les intersections induites par PoBOC sont raisonnables comparativement à la version modifiée de k-moyennes : PoBOC produit beaucoup moins d'intersections (coef. partition égal à 1.09 contre 1.66) et ces intersections ne bouleversent pas la qualité globale des clusters (Γ du même ordre que k-moyenne original).

En conclusion de cette analyse de PoBOC sur la base Iris, l'algorithme propose des résultats satisfaisants et sensiblement meilleurs que l'algorithme de référence "k-moyennes", tant du point de vue du nombre de clusters que de leur qualité et des intersections entre clusters. La section suivante rapporte les résultats obtenus sur d'autres bases de données de tailles et de natures variées.

2.4.2 Évaluation de PoBOC sur d'autres bases de données

Pour compléter l'analyse précédente, nous avons évalué l'algorithme PoBOC selon les mêmes indices sur quatre autres bases de données : Zoology, Wine, Soybean et Annealing également issues de l'UCI repository [122]. Ces bases ont été choisies de façon à évaluer les performances de l'algorithme dans diverses situations :

- le nombre d'objets varie de 101 (Zoology) à 898 (Annealing),
- le nombre de classes dans la classification de référence varie de 3 (Wine) à 19 (Soybean),
- les variables descriptives sont de différentes natures : numériques (Wine), numériques et symboliques (Annealing) ou uniquement symboliques (Zoology, Soybean).

Cette dernière caractéristique implique l'utilisation de mesures de dissimilarité adaptées : en l'absence de variables symboliques nous continuons à utiliser la distance euclidienne. Sinon, nous évaluons la proximité entre deux objets par la mesure proposée par Martin et Moal [120] et présentée dans le chapitre précédent.

Évaluation du nombre de clusters

Pour chacune des quatre bases de données, le schéma proposé par PoBOC est confronté, en terme de nombre de clusters, d'abord au nombre de classes dans la classification préétablie, ensuite au schéma issu de l'algorithme AutoClass et enfin aux schémas optimaux détectés sur les tracés des indices d'évaluation indépendants (ou relatifs). Cette étude comparative est présentée dans le tableau 2.5.

Base	Nb. objets	Nb. clusters proposés par			
		Classif. réf.	AutoClass	PoBOC	Évaluation indep.
Zoology	101	7	6 / **5**	**5**	3 / **5** / 7
Wine	178	3	3 / 2	**4**	**4** / 7
Soybean	683	19	7 / 5	**24**	9 / 15 / 17 / **24** / 26
Annealing	898	5	5	**17**	4 / 8 / 10 / **17** / 21

TAB. 2.5 – Recherche du nombre optimal de clusters sur les bases : Zoology, Wine, Soybean et Annealing.

On observe alors que les deux algorithmes PoBOC et AutoClass sont rarement en accord sur le nombre optimal de clusters, PoBOC proposant toujours davantage de clusters. Sur la base de la classification de référence, c'est l'algorithme AutoClass qui semble proposer les schémas adéquats (excepté pour la base Soybean). En revanche, lorsqu'on observe les schémas optimaux issus d'une analyse indépendante de toute connaissance extérieure, on retrouve systématiquement un schéma équivalent au résultat de PoBOC. Les tracés, à l'origine des valeurs présentées dans la dernière colonne du tableau, sont présentés en annexe B.

Le nombre de clusters proposé par PoBOC correspond donc à une organisation réelle et attestée des données. Si ce nombre semble parfois un peu élevé (par exemple pour la base Annealing), notons qu'il est préférable de proposer trop de clusters (faciles à fusionner par la suite) que trop peu (plus difficiles à diviser).

Statistique de *Huberts* Γ (externe)

Méthode	Zoology	Wine	Soybean	Annealing
k-moyennes	0.050±0.000	0.071±0.000	0.051±0.003	0.006±0.000
PoBOC sans recouv^{ts}	0.050	0.071	0.048	0.006
PoBOC	0.050	0.071	0.048	0.006

Statistique de *Rand*

Méthode	Zoology	Wine	Soybean	Annealing
k-moyennes	0.839±0.051	0.716±0.015	0.898±0.009	0.425±0.005
PoBOC sans recouv^{ts}	0.927	0.693	0.919	0.423
PoBOC	0.911	0.617	0.914	0.428

Statistique de *Huberts* Γ (interne)

Méthode	Zoology	Wine	Soybean	Annealing
k-moyennes	0.277±0.008	0.245±0.005	0.218±0.001	0.191±0.001
PoBOC sans recouv^{ts}	0.278	0.249	0.219	0.191
PoBOC	0.278	0.237	0.219	0.191
Classification de référence	*0.258*	*0.204*	*0.209*	*0.077*

TAB. 2.6 – Évaluation des schémas par des indices externes.

Évaluation des schémas comparativement aux classifications de référence

Les mesures proposées dans le tableau 2.6 concernent l'évaluation des schémas obtenus par les trois algorithmes : k-moyennes, PoBOC strict et PoBOC original, comparativement aux classifications proposées *a priori*. On note alors que l'indice d'évaluation externe Γ ne distingue ni les schémas stricts des schémas avec recouvrements, ni les partitions obtenues par PoBOC de celles obtenues par k-moyennes.

En revanche, la statistique de *Rand* présente un taux de liaisons (et non-liaisons) correctes à l'avantage de PoBOC sur les bases Zoology et Soybean. Sur Wine, les schémas obtenus par k-moyennes comptent plus de liaisons correctes que ceux obtenus par PoBOC. Enfin, sur la base Annealing les schémas sont plutôt équivalents du point de vue de cet indice. On remarque également par cet indice, que les liaisons en plus (et non-liaisons en moins) engendrées par les recouvrements entre clusters induisent une faible variation (positive ou négative) de cet indice. Autrement dit, il y a globalement une compensation entre la création de liaisons incorrectes et la suppression de non-liaisons incorrectes. Cette dernière remarque permet de justifier les multi-affectations proposées par PoBOC.

Enfin, le tableau inférieur montre que sur ces quatre jeux de données, les schémas de clustering proposés par k-moyenne et PoBOC correspondent mieux à la matrice de proximité calculée que la classification faisant référence. Comme pour la base Iris, cette observation nous encourage à poursuivre l'étude à partir d'indices d'évaluation indépendants (ou relatifs).

Évaluation de la qualité intrinsèque des schémas

En comparant les partitions strictes obtenues par k-moyennes et PoBOC (version stricte), on s'aperçoit que les clusters sont généralement plus compacts avec PoBOC ;

Inertie intra-cluster

Méthode	Zoology	Wine	Soybean	Annealing
k-moyennes	1.432±0.613	0.957±0.175	4.834±0.436	3.307±1.013
PoBOC sans recouv^{ts}	0.926	0.746	5.240	2.440
PoBOC	1.272	6.142	6.344	3.722

Inertie inter-clusters

Méthode	Zoology	Wine	Soybean	Annealing
k-moyennes	0.261±0.038	0.213±0.073	1.158±0.402	0.342±0.095
PoBOC sans recouv^{ts}	0.233	0.299	1.066	0.451
PoBOC	0.258	0.239	1.074	0.437

Statistique de *Huberts* Γ (relatif)

Méthode	Zoology	Wine	Soybean	Annealing
k-moyennes	0.090±0.008	0.087±0.002	0.052±0.002	0.036±0.003
PoBOC sans recouv^{ts}	0.094	0.089	0.050	0.040
PoBOC	0.094	0.078	0.050	0.039

Coefficient de partition

Méthode	Zoology	Wine	Soybean	Annealing
k-moyennes	1.00	1.00	1.00	1.00
PoBOC sans recouv^{ts}	1.00	1.00	1.00	1.00
PoBOC	1.08	1.38	1.15	1.21

TAB. 2.7 – Évaluation des schémas par des indices relatifs.

cette observation est vérifiée sur les bases Zoology, Wine et Annealing. Sur ce dernier jeu de données, la différence d'inertie intra-cluster est sensible et s'accompagne d'une dispersion entre clusters également meilleure pour le schéma issu de PoBOC. Cette double tendance est aussi vérifiée sur la base Wine pour laquelle les clusters proposés par PoBOC sont plus compacts et mieux séparés qu'avec k-moyennes. Sur les bases Zoology et Soybean, k-moyennes induit des schémas de partitionnement aux clusters assez bien séparés, contrairement à PoBOC.

Notons que le passage d'un schéma strict à un schéma avec recouvrements, *via* PoBOC, implique systématiquement une augmentation de l'inertie intra-cluster. Ceci s'explique par l'élargissement des clusters, conséquence des multi-affectations. En revanche, l'inertie inter-clusters n'en sort pas nécessairement diminuée. En effet, sur les bases Zoology et Soybean, les recouvrements permettent d'améliorer la dispersion entre clusters qui s'explique non pas par l'éloignement des clusters entre eux, mais par l'éloignement des clusters par rapport au centre de gravité de l'ensemble des objets.

Le dernier indice (coefficient de partition) confirme l'idée que la qualité d'une pseudo-partition s'évalue notamment par la quantité de recouvrements entre clusters. Ces intersections doivent rester "raisonnables" et l'excès peut engendrer une dégradation démesurée du schéma. C'est le cas pour le clustering proposé sur la base Wine : la valeur élevée[10] de ce coefficient nuit gravement à la qualité des clusters et notamment à leur compacité.

[10]Cette valeur doit être considérée relativement au faible nombre de clusters (4). Ainsi, un coefficient

2.5 Conclusion

Dans ce chapitre, nous avons présenté l'algorithme PoBOC (*Pole-Based Overlapping Clustering*), notamment en détaillant les trois principales étapes de traitement : la construction d'un graphe de dissimilarité, la construction des pôles et l'affectation multiple des objets à ces pôles. Nous avons montré que PoBOC est peu sensible à la présence d'outliers dans les données et qu'il génère des schémas de clustering comportant des classes de densité variable. Enfin, la définition d'un ensemble de "pôles" est fondamentale, puisqu'elle débouche sur le choix d'un nombre approprié de clusters et permet de déterminer précisément les affectations des objets restant.

Dans un deuxième temps, nous avons tenté d'analyser le traitement effectué par PoBOC sur un ensemble réel de données. Sur la base Iris, nous avons pu visualiser les processus de sélection d'objets, de construction des pôles et d'affectation des objets restants ; ces observations ont confirmé les caractéristiques attendues des pôles (ensembles homogènes et séparés). Cette évaluation visuelle a été complétée par une validation qualitative des schémas générés par PoBOC, relativement aux trois stratégies d'évaluation : externe, interne et relative. Il en résulte que PoBOC obtient globalement (sur l'ensemble des bases de données testées) de meilleurs résultats sur les tests externes et internes, que le schéma moyen obtenu par un algorithme du type k-moyennes. Sur l'évaluation relative, PoBOC obtient des résultats au moins équivalents, malgré la présence de recouvrements entre clusters.

Cette première phase expérimentale aide à cerner le fonctionnement ainsi que les performances de l'algorithme PoBOC. Cependant, elle n'est pas adaptée pour rendre compte de l'intérêt et de l'importance de construire des classes non-disjointes d'objets. Puisqu'il n'existe pas de mesure d'évaluation adaptée à ce type de schéma, nous proposons dans les chapitres suivants, de quantifier l'apport des intersections dans des applications qui utilisent le clustering comme outil de traitement. Pour chaque application, nous tâcherons de cerner les avantages ainsi que les limites liés à l'utilisation de pseudo-partitions par l'appréciation du résultat final de l'application (pouvoir prédictif des règles apprises, précision des classifieurs, etc.).

de partition de 1.38 pour un schéma à 4 clusters indique un taux de recouvrements bien supérieur à un schéma à 17 clusters, ayant pour coefficient de partition 1.21, tel que le schéma obtenu sur Annealing.

3

Apprentissage de règles de classification par regroupement d'objets

Sommaire

3.1 Introduction à l'apprentissage supervisé 80
3.2 Apprentissage d'un ensemble de règles 80
3.2.1 Le système AQ [123] 83
3.2.2 Le système CN2 [24] 83
3.2.3 Le système pFOIL [150] 84
3.2.4 Le système RISE [48] 84
3.3 Motivation de l'approche par regroupement 85
3.3.1 Le problème du choix d'un "bon" sélecteur 85
3.3.2 Utilisation du clustering dans un contexte supervisé 86
3.3.3 L'approche générale RuLe-Clust 87
3.3.4 Vers un pseudo-partitionnement des classes 88
3.4 Le système RuLe-Clust 88
3.4.1 Stratégie générale du classifieur 90
3.4.2 Test d'existence d'une règle unique 90
3.4.3 Algorithme de construction d'une règle 92
3.4.4 Variantes relatives à l'algorithme de clustering utilisé 92
3.5 Évaluation du système RuLe-Clust 95
3.5.1 Présentation de l'étude 95
3.5.2 Évaluation du principe général de décomposition 96
3.5.3 Évaluation de l'algorithme PoBOC 97
3.5.4 Influence d'une pseudo-partition 99
3.5.5 Comparaison avec d'autres classifieurs 100
3.6 Conclusion 101

3.1 Introduction à l'apprentissage supervisé

Contrairement aux deux chapitres précédents, nous nous situons à présent dans le cadre de l'apprentissage supervisé. Étant donné un ensemble d'exemples (ou observations) E, décrits sur un ensemble d'attributs $A = \{a_1, \ldots, a_p\}$, numériques et/ou symboliques, et étiquetés par une classe $c(e)$, on souhaite apprendre à prédire la classe d'un nouvel exemple e', sur la base des observations connues. E est alors appelé "ensemble des données d'entraînement".

Il existe différentes approches pour apprendre à classer à partir d'un ensemble d'entraînement ; on distingue notamment les méthodes basées sur les instances (IBL[1]) de celles basées sur des règles (RBL[2]).

L'apprentissage basé sur les instances est dominé par les approches du type k **Plus Proches Voisins** (kPPV). Pour un exemple à classer e', il s'agit de considérer l'ensemble des k exemples connus (appartenant à l'ensemble d'entraînement E) les plus proches de e' relativement à une mesure de (dis)similarité. La classe prédite pour ce nouvel exemple est alors la classe majoritaire, dans ce voisinage de k exemples. Les difficultés récurrentes, pour ce type de méthode, résident dans le choix du nombre k de voisins à considérer et dans la définition d'une mesure de (dis)similarité appropriée.

Dans ce travail, nous nous intéressons aux méthodes RBL. Nous avons cependant choisi d'introduire un exemple de méthode IBL en présentant l'approche des kPPV pour illustrer notre propos et parce que les expérimentations que nous envisageons dans la suite de ce chapitre, font référence à cette méthode de classification. Toutefois, le lecteur intéressé par les méthodes basées sur les instances trouvera une synthèse relativement complète de ce domaine d'étude dans [126].

3.2 Apprentissage d'un ensemble de règles

Dans cette section, nous nous intéressons aux méthodes de classification par apprentissage d'un ensemble de règles (RBL). Nous limitons notre étude au formalisme propositionnel, autrement dit, sans intégration de variables[3]. Les règles que nous cherchons à générer sont donc de la forme :

$$\text{Si } < condition > \text{ alors } Classe$$

où la *condition* (aussi appelée *complexe* ou *hypothèse*) est une conjonction de *sélecteurs*. Un *sélecteur* est de la forme $a_i \# v$ avec a_i un attribut de A, v appartient au domaine de valeurs de a_i, et $\#$ est un opérateur de l'ensemble $\{=, \neq, <, >, \leq, \geq\}$. On dira qu'un complexe (resp. un sélecteur) *couvre* un exemple, si le complexe (resp. le sélecteur) est vrai pour cet exemple. Ainsi, le complexe vide (conjonction de zero sélecteur) couvre tous les exemples de l'ensemble E.

La *classe* prédite par la règle correspond à la classe majoritaire de l'ensemble des exemples couverts par la condition de la règle. Finalement, la disjonction des règles apprises permet de couvrir l'ensemble des données d'entraînement. La prédiction de la classe d'un nouvel exemple e' est effectuée par la recherche d'une (ou plusieurs) règle(s) dont la condition couvre e'.

[1] *Instance-Based Learning*

[2] *Rule-Based Learning*

[3] L'extension à l'apprentissage de règles en logique du premier ordre est étudiée dans le chapitre 4.

Un ensemble de règles de classification peut être généré par différentes stratégies d'apprentissage, comme par exemple, les algorithmes génétiques, les arbres de décision ou les algorithmes de couverture séquentielle ou simultanée.

Les **algorithmes génétiques**, tels que le système GABIL [39], permettent d'apprendre des règles, codées sous forme d'une chaîne de bits. Par exemple, l'ensemble suivant (constitué de deux règles) :

$$R_1 : \text{Si } forme = cercle \ \wedge \ couleur = (rouge \vee vert \vee bleu) \text{ alors } classe = +$$
$$R_2 : \text{Si } forme = rectangle \ \wedge \ couleur = (rouge \vee vert) \text{ alors } classe = -$$

pourra être codé de la manière suivante :

	forme		couleur			classe	
	cercle	rectangle	rouge	vert	bleu	+	-
R_1	1	0	1	1	1	1	0
R_2	0	1	1	1	0	0	1

et donner lieu à la chaîne de bits suivante :

$$\underbrace{10\ 111\ 10}_{R_1}\ \ \underbrace{01\ 110\ 01}_{R_2}$$

Cette représentation est ensuite facilement utilisée par un opérateur génétique, pour effectuer successivement des modifications (croisement, mutation, etc.), contrôlées par une fonction de qualité, basée sur la précision du classifieur dérivé de l'ensemble de règles apprises.

Un **arbre de décision** peut également être interprété sous forme d'un ensemble de règles. ID3 [148] ou C4.5 [149] sont d'éminents exemples d'algorithmes de classification par construction d'arbres de décision. De façon très synthétique, il s'agit de considérer initialement l'ensemble des exemples E (racine), de rechercher l'attribut a^* le plus discriminant par rapport aux classes attendues, puis de générer autant de sous arbres que de modalités pour a^*. Récursivement, chaque sous-arbre est ensuite considéré indépendamment, jusqu'à aboutir à des sous-ensembles de données (feuilles) relativement purs.

Dans l'exemple présenté en figure 3.1, les données d'entraînement sont constituées de cinq exemples étiquetés par l'une des classes + ou − et décrits (tableau de gauche) sur deux attributs symboliques : *forme* et *couleur*. L'arbre de décision proposé (partie droite) peut donc être traduit en terme de règles, en considérant chaque chemin allant de la racine à une feuille. De l'exemple, on extrait les quatre règles suivantes :

$$R_1 : \text{Si } forme = rectangle \ \wedge \ couleur = rouge \text{ alors } classe = -$$
$$R_2 : \text{Si } forme = rectangle \ \wedge \ couleur = bleu \text{ alors } classe = +$$
$$R_3 : \text{Si } forme = rectangle \ \wedge \ couleur = vert \text{ alors } classe = -$$
$$R_4 : \text{Si } forme = cercle \text{ alors } classe = +$$

Dans un arbre de décision, chaque sous-arbre est dépendant de l'attribut choisi au nœud supérieur pour séparer les exemples. Ce choix est effectué à l'aide d'indices tels que l'indice de Gini (cf. annexe C), utilisé par exemple dans la méthode CART.

Dans la suite, nous nous intéressons plus particulièrement à une troisième stratégie

	forme	*couleur*	*classe*
e_1	cercle	bleu	+
e_2	cercle	rouge	+
e_3	rectangle	vert	-
e_4	rectangle	rouge	-
e_5	rectangle	bleu	+

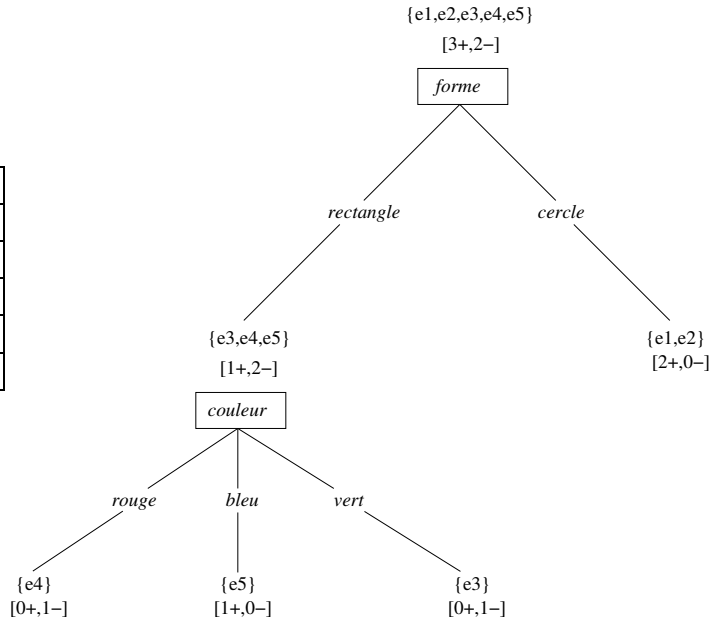


FIG. 3.1 – Exemple d'un arbre de décision.

d'apprentissage de règles de classification. Dans cette famille de systèmes, la méthode de construction d'un ensemble de règles diffère selon trois axes : la stratégie générale du classifieur, l'algorithme de couverture d'un ensemble d'exemples et enfin, la méthode de construction d'une règle.

De façon très générale, un classifieur peut adopter deux stratégies pour apprendre un ensemble de règles, à partir d'exemples étiquetés. Il est possible de considérer chaque classe l'une après l'autre et de construire un ensemble de règles propre à chacune des classes (apprentissage de concepts). Pour prédire l'appartenance d'un nouvel exemple e' à une classe c_i, on vérifie alors si e' est couvert ou non par au moins l'une des règles de couverture de c_i. L'autre approche consiste à générer l'ensemble de règles sans cibler de classe particulière. Le résultat prend alors la forme d'une liste de décision dans laquelle les règles sont ordonnées par pertinence décroissante. Prédire la classe d'une nouvelle instance e' revient à parcourir la liste dans l'ordre, et à retenir la première règle vérifiée par e', pour lui attribuer la classe correspondante.

L'algorithme de couverture d'un ensemble d'exemples permet d'appréhender des règles de façon séquentielle ou simultanée. Dans le premier cas, les règles sont construites les unes après les autres, en supprimant à chaque fois les exemples déjà couverts. Dans le cas d'une couverture simultanée, toutes les règles sont construites en même temps et indépendamment les unes des autres.

Enfin, on note deux principes de construction d'une règle : par généralisation ou par spécialisation. Le principe de généralisation considère un complexe très spécifique (e.g. la caractérisation d'un exemple) puis généralise ce complexe par suppression ou modification d'un sélecteur. A l'inverse, le processus de spécialisation, plus fréquent, considère une hypothèse très générale (e.g. le complexe vide), puis la spécialise par ajout de sélecteurs.

Nous nous appuyons sur les quatre systèmes d'apprentissage bien connus : AQ, CN2, pFOIL et RISE, pour illustrer ces différentes stratégies d'apprentissage.

3.2.1 Le système AQ [123]

Nous présentons, ci-dessous, le principe général du système AQ. Ce système fonctionne par apprentissage d'un ensemble de règles pour chacune des classes. L'algorithme de couverture d'une classe est séquentiel et chaque règle est construite par une approche du plus général au plus spécifique.

Stratégie du classifieur : Considérer chaque classe successivement. Pour une classe c_i donnée, diviser l'ensemble d'entraînement en deux sous ensembles : E^+ l'ensemble des exemples étiquetés c_i et E^- l'ensemble des exemples des autres classes. Exécuter l'algorithme de couverture sur E^+.

Algorithme de couverture : Générer une nouvelle règle jusqu'à ce que chaque exemple de E^+ soit couvert par au moins une règle. Les exemples positifs couverts par une règle générée précédemment sont supprimés de E^+.

Construction d'une règle : Choisir aléatoirement un exemple e de E^+ (non couvert), puis construire une `ETOILE` à partir de cet exemple. Une telle `ETOILE` est composée de l'ensemble des "meilleurs" complexes qui couvrent e et aucun exemple de E^-. Sélectionner le "meilleur" complexe de l'`ETOILE`. La notion de "meilleur complexe" est évaluée, entre autre, par une mesure de fréquence relative du nombre d'exemple positifs couverts par rapport au nombre total d'exemples couverts par le complexe (cf. annexe C).

Précisons que l'`ETOILE` est construire autour d'un exemple positif e^+, on considérant initialement le complexe vide (le plus général). Ensuite, un exemple négatif e^- est sélectionné, puis toutes les spécialisations possibles du complexe initial sont envisagées, de façon à rejeter e^- et couvrir toujours e^+. Ce processus de spécialisation est répété jusqu'à ce que tous les exemples négatifs soient rejetés.

Dans le système AQ, chaque règle générée dépend du choix de l'exemple à partir duquel l'"étoile" est construite. La difficulté est alors de choisir les exemples pertinents.

3.2.2 Le système CN2 [24]

Le système CN2 est assez proche du système AQ : les règles sont générées séquentiellement par une méthode du plus général au plus spécifique. Cependant, plutôt que de considérer chaque classe l'une après l'autre, le classifieur considère l'ensemble des exemples et recherche à chaque fois "la" meilleure règle.

Stratégie du classifieur : Considérer l'ensemble global E. Exécuter l'algorithme de couverture sur E.

Algorithme de couverture : Générer une nouvelle règle jusqu'à ce que chaque exemple de E soit couvert par au moins une règle. Les exemples couverts par une règle générée précédemment sont supprimés de E.

Construction d'une règle : La construction d'une règle se fait par recherche du meilleur complexe sur E. La procédure débute avec un ensemble de complexes candidats, réduit initialement à un unique complexe (le complexe vide). Ensuite, tous les sélecteurs sont envisagés de façon à générer autant de nouveaux complexes plus

spécifiques. Seuls les meilleurs sont retenus pour une nouvelle étape de spécialisation. Finalement, on arrête le processus lorsque l'un des complexes obtenus est discriminant pour l'une des classes (relativement à un critère spécifié par l'utilisateur, cf. annexe C).

CN2 produit alors une liste de décision, c'est à dire une liste de règles, ordonnées suivant l'ordre dans lequel elles ont été générées. Cet ordre correspond en quelque sorte à un ordre de pertinence, permettant de confronter un nouvel exemple aux règles pertinentes d'abord.

3.2.3 Le système pFOIL [150]

En logique du premier ordre, le système FOIL (*First-Order rule Induction*) [150] est connu pour ses capacités à générer un ensemble de clauses (ou règles) définissant un concept cible. Le principe général de l'algorithme peut être adapté au formalisme propositionnel. Cette variante nommée pFOIL (*propositional FOIL*) est résumée ci-dessous.

Stratégie du classifieur : Considérer chaque classe successivement. Pour une classe c_i donnée, diviser l'ensemble d'entraînement en deux sous ensembles : E^+ l'ensemble des exemples étiquetés c_i et E^- l'ensemble des exemples des autres classes. Exécuter l'algorithme de couverture sur E^+.

Algorithme de couverture : Générer une nouvelle règle jusqu'à ce que chaque exemple de E^+ soit couvert par au moins une règle. Les exemples positifs couverts par une règle générée précédemment sont supprimés de E^+.

Construction d'une règle : Considérer initialement le complexe vide (couvre toutes les instances de E). Ajouter itérativement les sélecteurs jusqu'à ce que la règle ne couvre aucun exemple de E^-. Les sélecteurs ajoutés maximisent la mesure de gain d'information (cf. annexe C).

On note que pFOIL procède de façon analogue au système AQ, en apprenant un ensemble de règles pour chaque classe, successivement. L'algorithme de couverture d'une classe d'exemples est également fondé sur le même principe ("glouton") que AQ et CN2. En revanche, plutôt que de choisir un exemple et de dériver une "étoile" (comme c'est le cas dans AQ), pFOIL utilise une technique de construction de règle, similaire au système CN2, c'est à dire par spécialisation du complexe le plus général (complexe vide).

Remarquons que les trois systèmes précédents procèdent par couverture séquentielle (algorithme "glouton") d'un ensemble d'exemples et construisent chaque règle avec une approche du plus général au plus spécifique.

3.2.4 Le système RISE [48]

Le système RISE diffère des trois stratégies précédentes, notamment par une méthode de couverture "simultanée" d'un ensemble d'exemples. La différence porte également sur la technique de construction d'une règle par une approche, cette fois, du plus spécifique au plus général. Nous présentons la stratégie globale de RISE :

Stratégie du classifieur : Considérer l'ensemble global E. Exécuter l'algorithme de couverture de E.

Algorithme de couverture : Générer toutes les règles simultanément, en considérant initialement l'ensemble des complexes spécifiques à chaque exemple de E (chaque complexe couvre un exemple distinct), puis par étapes successives de généralisation de chaque complexe et suppression des doublons ou équivalents. Un complexe généralisé ne sera conservé que s'il améliore la qualité globale de l'ensemble des complexes, relativement à une mesure de qualité (cf. annexe C).

3.3 Motivation de l'approche par regroupement

3.3.1 Le problème du choix d'un "bon" sélecteur

Dans les systèmes présentés précédemment, le problème du choix d'un "bon" sélecteur se pose. En effet, que l'on choisisse de construire une règle par spécialisation ou généralisation, il s'agit de choisir "le" sélecteur le plus pertinent (à ajouter ou à supprimer) en évaluant la règle induite par cette modification. Ainsi, dans le système pFOIL par exemple, la règle est construite à partir du complexe vide, en y ajoutant successivement un sélecteur, maximisant le gain d'information, jusqu'à ce que la règle rejette tous les exemples étiquetés comme instances négatives de la classe cible.

Le critère de gain d'information, ainsi que d'autres mesures traditionnellement utilisées, telles que la mesure d'entropie, l'indice de Gini ou encore les tests du χ^2 ou de Laplace, aident à choisir un sélecteur pertinent, c'est à dire permettant de séparer au mieux les exemples positifs des exemples négatifs.

Cependant, il existe de nombreux cas dans lesquels ces mesures ne parviennent pas à dégager "le/les" sélecteur(s) approprié(s). Les choix effectués influencent fortement la suite du processus et risquent d'entraîner la construction d'une règle peu pertinente. Dans l'exemple présenté en figure 3.2, les sélecteurs $x_1 < 2$ et $x_1 > 4$ maximisent le gain d'information (≈ 0.18).

Par le système pFOIL, l'algorithme de couverture de la classe "+" pourrait conduire aux quatre règles suivantes :

R_1 : Si $x_1 < 2$ alors $classe = +$
R_2 : Si $x_1 > 4$ alors $classe = +$
R_3 : Si $x_2 < 2 \ \wedge \ x_1 < 3$ alors $classe = +$
R_4 : Si $x_2 > 4 \ \wedge \ x_1 > 2$ alors $classe = +$

Pourtant, il apparaît clairement que deux règles suffisent à couvrir l'ensemble des instances positives (e.g. les complexes $x_1 < 4 \ \wedge \ x_2 < 3$ et $x_1 > 2 \ \wedge \ x_2 > 3$). Ces deux règles correspondent chacune à la couverture d'une sous-classe de la classe cible. On note également que ces sous-classes sont constituées d'exemples "proches", c'est à dire à valeurs assez similaires sur les mêmes attributs. Dans la suite de cette section, nous proposons une solution (partielle) au problème de choix d'un "bon" sélecteur. Il s'agit d'apporter une connaissance supplémentaire concernant l'organisation interne d'une classe en sous-classes. Cette connaissance - pouvant être apprise par une méthode de classification non-supervisée - permettra de considérer différents sous-problèmes correspondant chacun à la couverture d'une sous-classe de la classe cible.

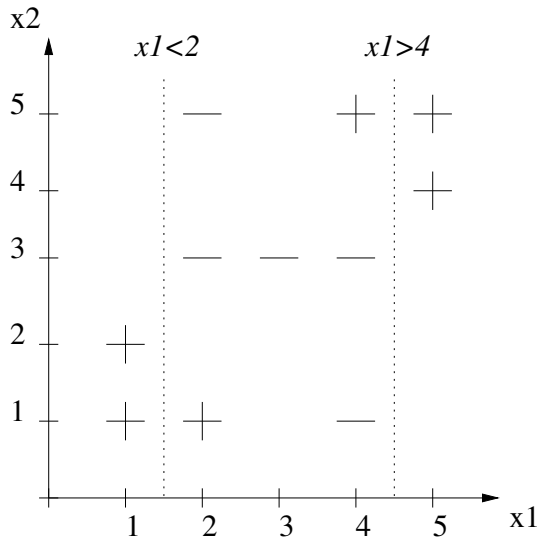


FIG. 3.2 – Un exemple de choix de sélecteur.

3.3.2 Utilisation du clustering dans un contexte supervisé

Certaines études ont déjà présenté l'intérêt d'utiliser un processus de clustering pour améliorer la qualité d'un classifieur, notamment dans des systèmes d'apprentissage basés sur les instances (IBL).

En effet, la classification par une approche du type kPPV nécessite, pour chaque nouvelle instance, de la comparer à toutes les données d'entraînement. Lorsque l'ensemble d'entraînement est de taille très importante, le calcul des (dis)similarités peut devenir très coûteux. Une manière de réduire ce coût est d'effectuer un partitionnement de chaque classe et de considérer uniquement les représentants de chaque cluster comme ensemble de données d'entraînement [35].

Dans le cas de la classification par fonctions radiales de base, les centres des gaussiennes, utilisées comme fonctions "noyaux", peuvent également être déterminés par les centres des clusters, issus d'un processus de partitionnement [128].

Plus récemment, [182] se sont intéressés à l'utilisation des méthodes de clustering pour la recherche d'un modèle approprié de fonctions de séparation. Ainsi, chaque classe cible donne lieu à plusieurs fonctions, de complexité différente.

Les méthodes de clustering, utilisées dans les trois exemples précédents, sont respectivement : l'algorithme des k-moyennes, les cartes auto-organisatrices (SOM) et l'algorithme EM[4]. On remarque alors que ces trois méthodes nécessitent de fournir le paramètre k, correspondant au nombre de clusters à générer. Ce paramètre est obtenu en itérant le processus pour différentes valeurs de k, et en observant la performance du classifieur induit. Ce procédé augmente sensiblement la complexité du système global.

Fort de cette dernière observation, et constatant l'absence d'études de ce genre dans le cas RBL, nous proposons d'étudier le comportement de l'algorithme PoBOC dans un processus d'apprentissage de règles, par décomposition des classes.

[4]Dans l'étude citée, les partitions floues résultantes de l'algorithme EM, sont restreintes à des partitions strictes, par affectation de chaque objet au cluster le plus proche.

3.3.3 L'approche générale RuLe-Clust

Le système général RuLe-Clust (*Rule-Learning from Clustering*) , que nous avons développé, est fondé sur une méthode du type pFOIL. Chaque classe est considérée l'une après l'autre, et un ensemble de règles est appris sur chacune des classes. Ces règles sont générées par une approche du plus général au plus spécifique. La différence avec le système pFOIL concerne la stratégie utilisée pour couvrir les exemples de chaque classe. Plutôt que de considérer tous les exemples de la classe, et d'apprendre les règles par un algorithme glouton, nous proposons de décomposer la classe en plusieurs sous-classes (clusters), puis d'apprendre une règle pour chaque sous-classe obtenue.

Nous présentons, ci-dessous, le principe général du système RuLe-Clust :

Stratégie du classifieur : Considérer chaque classe successivement. Pour une classe c_i donnée, diviser l'ensemble d'entraînement en deux sous-ensembles : E^+ l'ensemble des exemples étiquetés c_i et E^- l'ensemble des exemples des autres classes. Effectuer une décomposition de E^+ en sous-classes $E_1^+, \ldots, E_k^+$ telles que chaque ensemble E_i^+ puisse être couvert par une seule règle, rejetant tous les exemples de E^-.

Algorithme de couverture : Apprendre simultanément une règle par sous-classe, en considérant les sous-problèmes suivants : E_i^+ est l'ensemble des exemples positifs et E^- l'ensemble des exemples négatifs.

Construction d'une règle : Considérer initialement le complexe vide (couvre toutes les instances de E). Ajouter itérativement les sélecteurs jusqu'à ce que la règle ne couvre aucun exemple de E^-. Le sélecteur ajouté est celui qui, parmi les sélecteurs qui couvrent tous les exemples de E_i^+, rejette le plus d'exemples négatifs (non-encore rejetés).

Sur l'exemple présenté en figure 3.2, un algorithme traditionnel de clustering proposera une décomposition de l'ensemble E^+ (matérialisé sur la figure par les "+") en deux sous-classes E_1^+ et E_2^+, comme indiqué sur la figure 3.3 (à gauche). Les deux sous-classes E_1^+ et E_2^+ donnent lieu respectivement aux sous-problèmes P_1 (centre) et P_2 (droite).

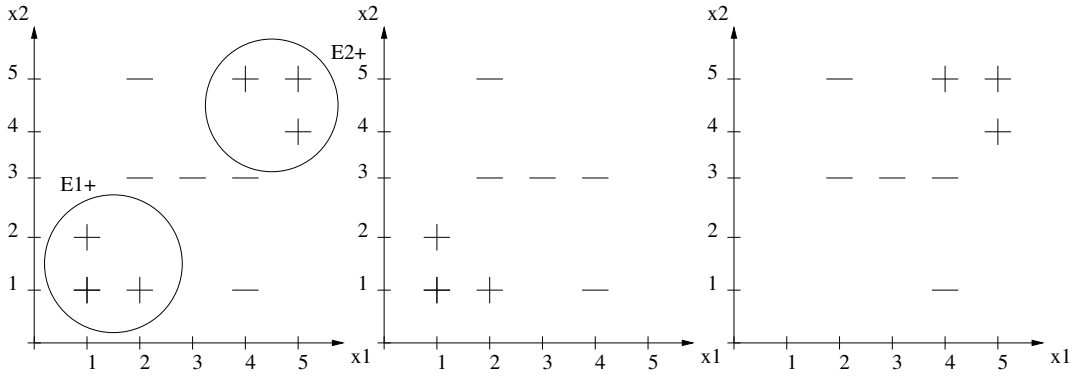


FIG. 3.3 – Exemple de décomposition d'une classe.

Sur le problème P_1, partant du complexe vide, l'ensemble des sélecteurs candidats C est constitué des sélecteurs qui couvrent les trois exemples positifs.

$$C = \{x_1 < 3, x_1 < 4, x_1 < 5, \ldots, x_2 < 3, x_2 < 4, x_2 < 5, \ldots\}$$

Parmi ces sélecteurs, $x_2 < 3$ est celui qui rejette le plus d'exemples négatifs (4 exemples négatifs rejetés sur 5), il est donc choisi pour spécialiser le complexe. Parmi les sélecteurs restant, deux sélecteurs permettent de rejetter le dernier exemple négatif, $x_1 < 3$ et $x_1 < 4$. Par défaut nous choisissons d'ajouter le sélecteur le plus général. La règle apprise est donc :

$$R_1 : \text{ Si } x_2 < 3 \wedge x_1 < 4 \text{ alors } classe = +$$

Par un procédé analogue, le sous-problème P_2 permet de dégager la règle suivante :

$$R_2 : \text{ Si } x_2 > 3 \wedge x_1 > 2 \text{ alors } classe = +$$

3.3.4 Vers un pseudo-partitionnement des classes

Nous venons de présenter l'intérêt d'organiser les données d'une classe par partitionnement, avant de chercher à la couvrir par un ensemble de règles. Dans les chapitres précédents, nous avons également avancé l'idée que les schémas de clustering avec intersections entre les clusters apportent une richesse d'information supplémentaire, comparativement aux schémas stricts traditionnels. Nous pouvons alors nous demander si l'utilisation d'une méthode de pseudo-partitionnement plutôt qu'un algorithme classique de partitionnement strict, peut avoir une influence sur la nature ou sur la qualité des règles apprises.

Tout d'abord, on peut noter que, dans le système RuLe-Clust, même si l'algorithme de clustering utilisé produit un ensemble de sous-classes disjointes $\{E_1^+, \ldots, E_k^+\}$, il est probable que les règles $R_1, \ldots, R_k$ apprises à partir de ces sous-classes, ne soient pas disjointes (l'intersection des complexes n'est pas vide).

Dans l'exemple présenté en figure 3.4, les deux clusters disjoints E_1^+ et E_2^+ (figure de gauche), conduisent aux deux règles R_1 et R_2 (figure de droite). Or les deux sous-ensembles, correspondant aux couvertures respectives des règles, s'intersectent. On pourrait donc considérer, à partir de cet exemple, que le schéma de partitionnement obtenu n'est pas déterminant, et que l'étape de construction des règles corrige les imperfections de ce schéma, notamment par élargissement des clusters à des sous-classes non-disjointes.

Pourtant, la méthode de construction d'une règle de couverture d'une sous-classe, ne permet pas systématiquement de corriger le schéma de clustering proposé. En effet, l'ajout d'un exemple négatif à l'exemple précédent (cf. figure 3.5), peut influencer le choix des sélecteurs et restreindre une règle. Dans la figure 3.5 (figure de droite), le sélecteur $x_2 > 4$ permet de rejeter le maximum d'exemples négatifs et risque donc d'être choisi en premier pour couvrir E_2^+.

En revanche, l'utilisation d'une méthode produisant un pseudo-partitionnement (cf. figure 3.6), oblige le système à générer, lorsque cela est possible, les règles correspondant à l'organisation réelle des exemples de la classe. Le partitionnement présenté en figure 3.6 (figure de gauche), supprime de la liste de candidats pour la couverture de E_2^+, le sélecteur $x_2 > 4$, qui ne couvre pas toutes les instances de E_2^+.

En conclusion, le recours à une méthode de clustering autorisant les recouvrements entre classes peut aider à générer des règles correspondant à des sous-classes naturelles, c'est à dire proches de l'organisation réelle des exemples de la classe.

3.4 Le système RuLe-Clust

Dans cette section, nous présentons de façon détaillée le système RuLe-Clust. Le principe du classifieur est défini, avant de proposer les procédures de test d'existence et de

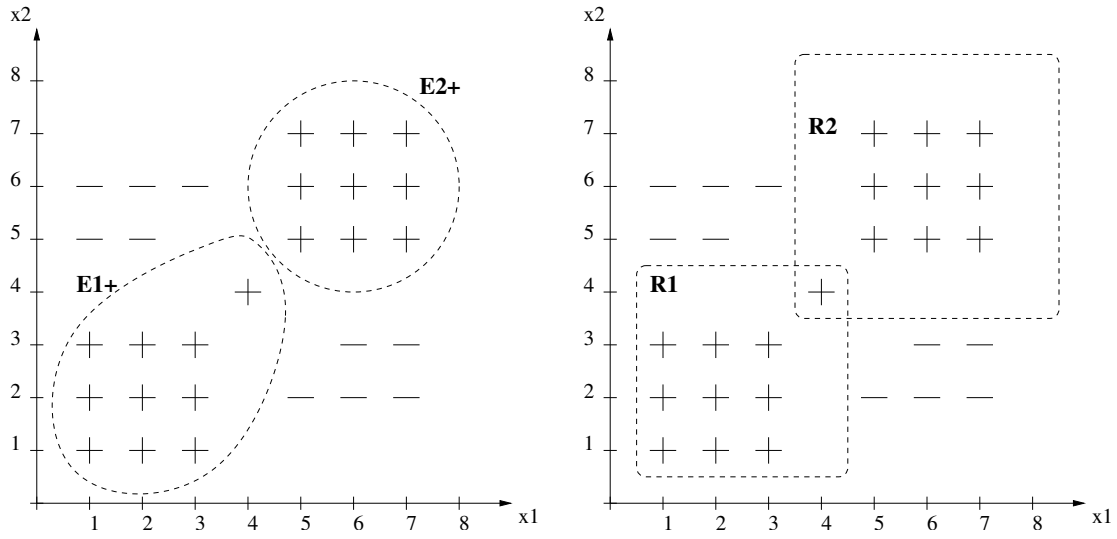


FIG. 3.4 – Exemple de partitionnement disjoint conduisant à des règles non-disjointes.

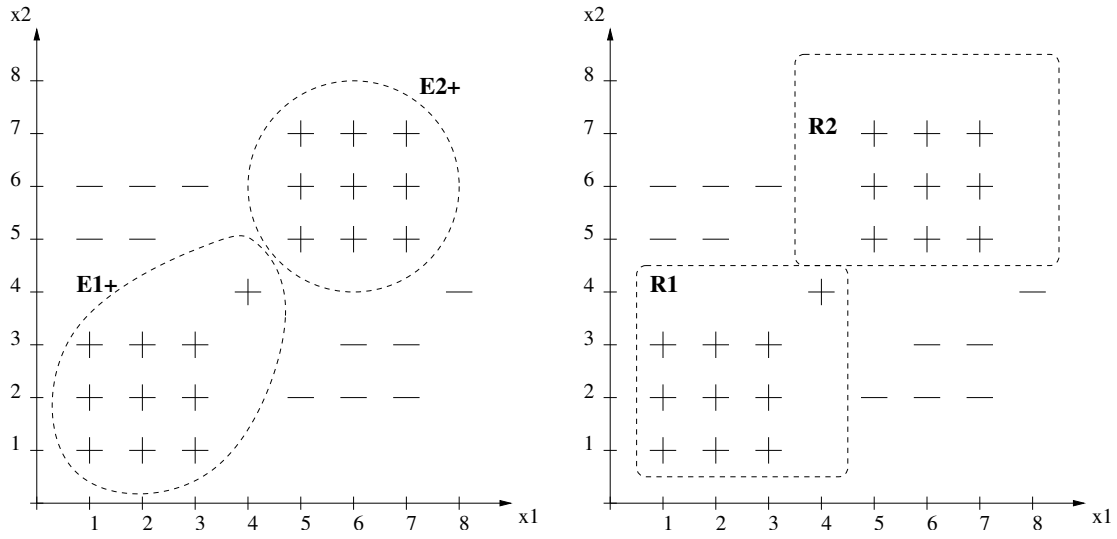


FIG. 3.5 – Exemple de partitionnement disjoint conduisant à des règles disjointes.

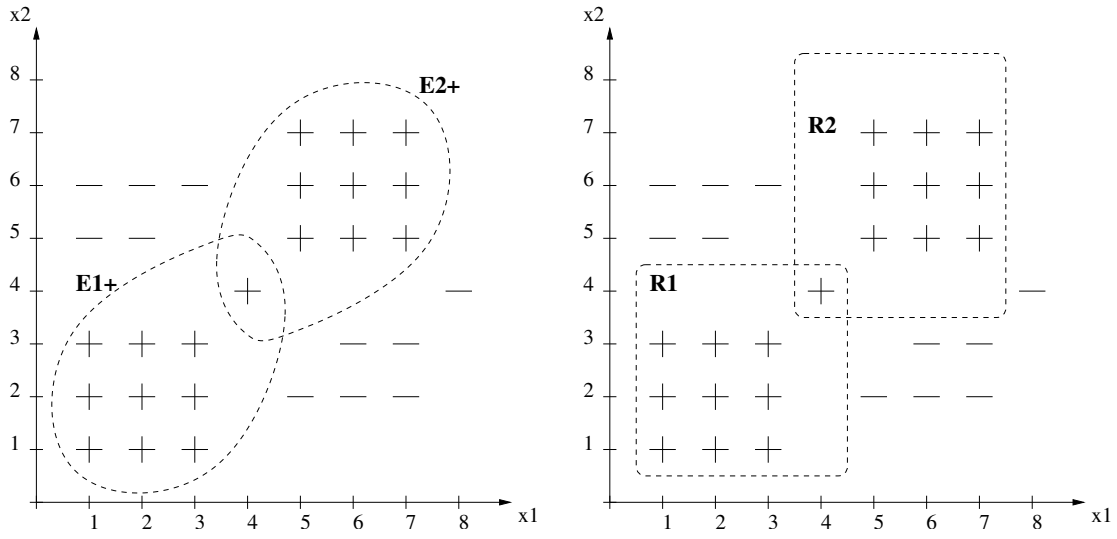


FIG. 3.6 – Exemple de partitionnement non-disjoint conduisant à des règles non-disjointes.

construction d'une règle. Nous terminons en précisant les variantes envisagées pour l'utilisation de différentes méthodes de clustering.

3.4.1 Stratégie générale du classifieur

La procédure générale du système RuLe-Clust, présentée en figure 3.7, reprend la stratégie classique des systèmes tels que AQ ou pFOIL. Chaque classe d'exemples est considérée l'une après l'autre, donnant lieu chacune à un ensemble de règles, par appel de la procédure `COUVRIR-RC` (`COUVRIR` par construction de `R`ègles orientées par `C`lustering).

Notons que le système RuLe-Clust commence par calculer la matrice des (dis)similarités sur E. Cette matrice sera utilisée par la suite, lors du processus de décomposition.

La fonction `COUVRIR-RC` est définie en figure 3.8. Pour une classe donnée, on commence par tester l'existence d'une règle discriminante pour la classe. Si une telle règle existe, elle est générée puis renvoyée en tant que règle unique. Dans le cas contraire (cas le plus fréquent), le processus de décomposition s'exécute.

Le processus de décomposition consiste à partitionner la classe d'exemples en k clusters, par appel d'un algorithme de clustering. Chaque cluster est alors soumis au test d'existence d'une règle discriminante pour ce cluster, le cas échéant, la règle est générée et stockée dans l'ensemble de règles à renvoyer. Tant qu'il n'est pas vide, l'ensemble des exemples non-couverts est redécomposé par ce même procédé. Dans le cas où aucun cluster n'a permis de construire une règle, chaque cluster est redécomposé indépendamment.

3.4.2 Test d'existence d'une règle unique

L'existence d'une règle unique est une contrainte fondamentale dans le système RuLe-Clust. Ce test permet à la fois de vérifier la cohérence d'un cluster et d'accélérer le processus de décomposition.

En effet, l'existence d'une règle unique, séparant l'ensemble des exemples d'un cluster donné de l'ensemble des exemples négatifs, signifie que les exemples du cluster partagent

Procédure `RuLe-Clust`(E) :

Soit E l'ensemble des exemples et $c_1, \ldots, c_p$ les étiquettes de classe sur E,

Soit R l'ensemble de règles. Initialement $R = \emptyset$
Construire la matrice de (dis)similarité S sur E,

Pour chaque c_i $(i = 1 \ldots p)$:

Soit E^+ l'ensemble des exemples étiquetés c_i dans E,
Soit E^- l'ensemble des exemples étiquetés c_j $(j \neq i)$ dans E,

Appeler la procédure `COUVRIR-RC`(E^+, E^-, S) pour générer k règles $r_1, \ldots, r_k$,
Ajouter $r_1, \ldots, r_k$ à R.

Retourner R.

FIG. 3.7 – Procédure principale du système RuLe-Clust.

Fonction `COUVRIR-RC`(E^+, E^-, S) :

Soit E^+ l'ensemble des exemples positifs (à couvrir),
Soit E^- l'ensemble des exemples négatifs,
Soit S une matrice de (dis)similarité sur E^+,

Si `EXISTE-REGLE-UNIQUE`(E^+, E^-) **alors**
Appeler `CONSTRUIRE-REGLE(`E^+, E^-`)` pour générer une règle r,
Retourner r,

Sinon
Appeler `DECOMPOSE`(E^+, E^-, S) pour générer k clusters $E_1^+, \ldots, E_k^+$,
Soit R_1 la disjonction des règles associées aux groupes pour lesquels il existe
une règle,
Soit E' l'ensemble des exemples positifs non-couverts par R_1,
Soit R_2 =`COUVRIR-RC`(E', E^-, S),
Retourner la disjonction $R_1 \vee R_2$.

FIG. 3.8 – Algorithme de couverture d'une classe par clustering.

Procédure EXISTE-REGLE-UNIQUE(E^+, E^-) :

Soit E^+ l'ensemble des exemples positifs (à couvrir),
Soit E^- l'ensemble des exemples négatifs,

Soit H l'ensemble des sélecteurs qui couvrent tous les exemples de E^+,
Soit COMPLEXE, le complexe défini par la conjonction de tous les sélecteurs de H,

$$\texttt{COMPLEXE} = \bigwedge_{h_i \in H} h_i,$$

Si COMPLEXE rejette tous les exemples de E^- **alors Retourner** VRAI,
Sinon Retourner FAUX.

FIG. 3.9 – Test d'existence d'une règle unique.

certaines propriétés (au moins celles constituant le complexe de la règle). Ces similitudes justifient donc leur appartenance commune à un cluster.

Par ailleurs, le test de couverture, tel qu'il est présenté en figure 3.9, est de complexité sensiblement inférieure à l'algorithme de construction d'une règle. Ainsi, plutôt que de lancer un processus d'apprentissage de règle, pouvant ne pas aboutir, nous proposons de tester préalablement, et rapidement, l'existence de cette règle. Ce test consiste à générer tous les sélecteurs possibles (de la forme *attribut#valeur*), si la conjonction de l'ensemble des sélecteurs rejette tous les exemples négatifs, alors il existe une règle discriminante.

3.4.3 Algorithme de construction d'une règle

L'algorithme de construction d'une règle (figure 3.10) utilise, pour couvrir un ensemble E^+, une approche du plus général au plus spécifique, inspirée du système pFOIL. Partant du complexe vide, soit le plus général, ce dernier est progressivement spécialisé par ajout du meilleur sélecteur parmi ceux couvrant l'ensemble E^+. Le "meilleur" sélecteur est celui qui rejette le plus de contre exemples, parmi ceux non rejetés jusqu'alors. Lorsque tous les contre exemples sont rejetés par au moins l'un des sélecteurs constituant le complexe, ce complexe est retourné.

Cette méthode de construction permet d'approximer la règle discriminante (relativement aux deux ensembles E^+ et E^-) la plus générale.

3.4.4 Variantes relatives à l'algorithme de clustering utilisé

La procédure COUVRIR-RC, présentée en figure 3.8, propose une stratégie générale pour la décomposition d'une classe en sous-classes pour lesquelles il existe une règle unique, discriminante. Cependant, selon la méthode de clustering utilisée, il faudra adapter le processus de décomposition afin de tenir compte du paramétrage (nombre de clusters attendu) et du type de schéma retourné (partition ou hiérarchie) par l'algorithme. Dans la suite, nous énumérons ces variantes pour les algorithmes suivants : méthode des k-moyennes, méthode hiérarchique agglomérative et PoBOC.

Procedure CONSTRUIRE-REGLE(E^+, E^-) :

Soit E^+, l'ensemble des exemples positifs (à couvrir),
Soit E^-, l'ensemble des exemples négatifs,

Soit H, l'ensemble des sélecteurs qui couvrent tous les exemples de E^+,
Soit COMPLEXE, le complexe vide,

Tant que E^- n'est pas vide :

> Soit h, le sélecteur de H qui rejette le plus d'exemples négatifs,
> COMPLEXE=COMPLEXE$\wedge$ h,
> Supprimer de E^-, les exemples non-couverts par h,
> $H = H \setminus h$.

Retourner COMPLEXE.

FIG. 3.10 – Heuristique de construction d'une règle.

Variante pour l'algorithme des k-moyennes

Comme nous l'avons observé dans le chapitre 1, l'algorithme des k-moyennes ainsi que la plupart de ses variantes, nécessite de paramétrer à l'avance, le nombre k de clusters désiré. Dans certaines applications en classification non-supervisée, la connaissance des données par l'utilisateur ou l'aide d'un expert permet d'estimer ce paramètre. En revanche, dans l'application visée ici, il est difficile d'estimer à l'avance le nombre approprié de sous-classes.

Nous choisissons alors d'adapter la procédure COUVRIR-RC de la façon suivante : on commence par décomposer la classe en 2 clusters (k-moyennes pour $k = 2$). Si aucun des deux clusters ne permet de construire une règle discriminante, on tente une nouvelle décomposition de la classe en 3 clusters (k-moyennes pour $k = 3$). On réitère le procédé en augmentant k, jusqu'à ce que l'un des clusters, au moins, donne lieu à une règle discriminante.

Chaque fois qu'un cluster est "discriminé" par une règle, on considère l'ensemble des exemples non-couverts puis on exécute à nouveau la procédure de décomposition sur cet ensemble à partir de $k = 2$. Cette procédure est stoppée lorsque tous les clusters générés sont "discriminés". Le schéma présenté en figure 3.11 illustre la procédure de décomposition.

Variante pour les méthodes agglomératives hiérarchiques

Dans le cas où la décomposition des classes est effectuée par une méthode hiérarchique, la procédure est simplifiée. Le dendrogramme est généré une seule fois, puis il est parcouru de la racine vers les feuilles avec un test d'existence d'une règle discriminante sur chaque nœud. Si, pour un nœud donné, les exemples contenus dans ce nœud ne peuvent conduire à une règle discriminante, la procédure est appelée récursivement sur chacun de ces nœuds fils.

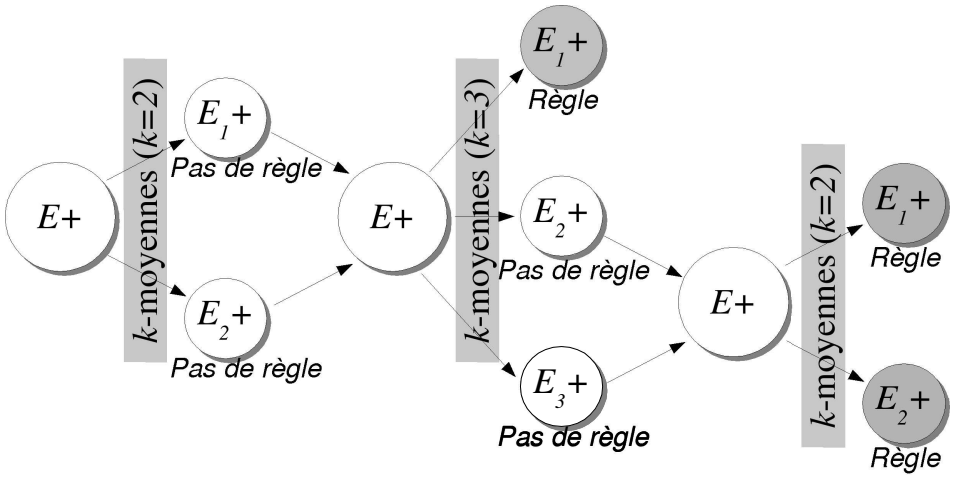


FIG. 3.11 – Processus de décomposition pour l'algorithme des k-moyennes.

Variante pour l'algorithme PoBOC

L'utilisation de l'algorithme PoBOC (cf. chapitre 2), présente l'avantage de ne pas avoir à proposer le paramètre k. Ainsi, pour une classe donnée, PoBOC renvoie une pseudo-partition de la classe en k clusters, k étant déterminé automatiquement. Contrairement à l'adaptation proposée pour l'utilisation de k-moyennes, le processus de décomposition utilisant PoBOC n'impliquera pas de redécompositions successives (et coûteuses) d'un même ensemble d'exemples.

En revanche, les trois situations suivantes sont à envisager :

Aucun cluster n'induit de règle discriminante : Ce cas, possible en théorie, se produit rarement en pratique. Le cas échéant, la procédure de décomposition est de nouveau appelée indépendamment sur chacun des clusters.

Certains clusters seulement induisent une règle discriminante : La procédure de décomposition est appelée sur l'ensemble des exemples n'appartenant à aucun des clusters couverts.

Tous les clusters induisent une règle discriminante : On cherche alors à limiter le nombre de règles[5]. Pour cela on organise hiérarchiquement les clusters de façon à

[5]Trop de règles pourrait induire un phénomène de sur-adéquation aux données d'entraînement (*overfitting*).

construire un dendrogramme dont les feuilles correspondent aux clusters obtenus (algorithme agglomératif hiérarchique du lien simple). On utilise ensuite la variante de décomposition adaptée au cas hiérarchique.

3.5 Évaluation du système RuLe-Clust

3.5.1 Présentation de l'étude

Objectifs de l'étude

Les expérimentations, reportées dans ce qui suit, permettent de répondre aux trois interrogations suivantes :

(1) Dans quelle mesure la décomposition des classes permet-elle d'améliorer le pouvoir prédictif des règles apprises ?

(2) L'algorithme de classification non-supervisée PoBOC est-il adapté à cette tâche de décomposition ?

(3) Si oui, est-ce lié au fait que PoBOC partitionne les données en classes non-disjointes ?

Pour répondre à chacune de ces questions, nous avons effectué les comparaisons appropriées. Dans des situations identiques, nous observons d'abord les résultats obtenus avec ou sans décomposition (pFOIL vs. RuLe-Clust). Ensuite, nous proposons une étude comparative de la performance du système RuLe-Clust pour différentes méthodes de décomposition (algorithmes de clustering différents). Enfin, nous présentons la qualité des classifieurs induits par deux méthodes de décomposition comparables, l'une proposant un partitionnement strict, l'autre un pseudo-partitionnement.

Afin de cerner les capacités globales du système RuLe-Clust, nous situons finalement notre système par rapport à deux méthodes traditionnelles de classification supervisée : la classification par construction d'un arbre de décision (C4.5 [149]) ainsi qu'une approche basée sur les instances (1 Plus Proche Voisin).

Présentation des données

Les expérimentations sont effectuées sur 10 ensembles de données provenant de l'UCI repository [122]. Les bases sélectionnées sont choisies de façon à traiter différents types de données (attributs numériques et/ou symboliques), de tailles variées (entre 100 et 700 exemples), et organisées en plus ou moins de classes (*e.g.* Credit : 690 exemples organisés en deux classes ; Audiology : 226 exemples organisés en 24 classes). Le tableau 3.1 résume les caractéristiques de chacun des 10 ensembles de données choisis.

Méthode d'évaluation

Tous les résultats présentés par la suite sont obtenus dans les mêmes conditions expérimentales. Autrement dit, les méthodes sont comparées sur les mêmes échantillons d'entraînement et de test ainsi que sur la même matrice de (dis)similarité.

Lorsqu'un découpage "ensemble d'entraînement/ensemble de test" est prédéfini (*e.g.* dans le cas des bases Audiology et Soybean), l'évaluation sur cette base consiste à exécuter

Base	Nb. exemples	Nb. Classes	Nb. attributs		Testset
			numériques	symboliques	fourni
AUDIOLOGY	226	24	0	69	oui
CREDIT	690	2	6	10	non
GLASS	214	6	10	0	non
HEART DISEASE	123	2	13	0	non
HEPATITIS	155	2	6	13	non
IRIS	150	3	4	0	non
SOYBEAN	683	19	0	35	oui
THYROID	216	3	5	0	non
WINE	178	3	13	0	non
ZOOLOGY	101	7	1	16	non

TAB. 3.1 – Présentation des ensembles de données.

10 fois le processus de classification sur ce découpage[6]. La moyenne sur ces 10 exécutions tient lieu de résultat.

Lorsqu'aucun découpage particulier n'est proposé, l'évaluation sur la base est effectuée par validations croisées (découpage choisi : ensemble d'entraînement 90%, ensemble de test 10%). On execute alors 5 validations croisées, le résultat reporté correspond à la précision moyenne sur les 5 executions. Nous proposons également l'écart type pour chaque résultat, cette valeur correspond à la moyenne des écarts types pour chaque validation croisée. Cette information complémentaire permet d'évaluer les variations de performances d'un échantillon d'entraînement à un autre (robustesse du classifieur).

Les expérimentations du système RuLe-Clust nécessitent le calcul d'une matrice de (dis)similarité. Comme pour le chapitre précédent, nous avons choisi d'utiliser la mesure proposée dans [120]. Cette mesure repose sur la définition d'un langage de descriptions dont un sous-ensemble de taille fixée est généré aléatoirement. Ce langage est différent pour chaque exécution (chaque validation croisée). Ce choix nous permet de traiter indifféremment les données décrites par des attributs numériques et/ou symboliques.

Pour synthétiser les résultats présentés dans chaque tableau de comparaison, nous proposons systématiquement en dernière ligne, le "rang moyen" de la méthode utilisée, par rapport aux autres méthodes évaluées. Ainsi, chaque résultat est annoté (nombre en exposant) par un numéro. Par exemple, l'annotation 83.4 ± 3.6^2 signifie qu'en moyenne, la méthode utilisée parvient à classer correctement 83.4% des exemples de l'ensemble de test, avec un écart type moyen de 3.6 (variations dans une même validation croisée), et que cette valeur correpond au deuxième meilleur résultat sur cette base et dans cette étude (parmi les valeurs de la même ligne et du même tableau).

3.5.2 Évaluation du principe général de décomposition

Pour évaluer de façon générale le système RuLe-Clust, nous devons comparer la qualité d'un ensemble de règles généré sans décomposition préalable des classes avec l'ensemble des règles apprises en utilisant la connaissance apportée par une décomposition.

[6]Des variations apparaissent d'une exécution à une autre, car la matrice de (dis)similarité est recalculée à chaque exécution.

RuLe-Clust et pFOIL utilisent le même algorithme de construction d'une règle, mais diffèrent dans la stratégie de couverture d'une classe. Pour une classe donnée, pFOIL propose une couverture séquentielle de la classe tandis que RuLe-Clust utilise les sous-classes obtenues par un algorithme de clustering. Nous choisissons alors de comparer ces deux systèmes. Les résultats de cette étude sont reportés dans le tableau 3.2.

Base	pFOIL	RuLe-Clust (PoBOC)
Audiology	61.5 ± 0.0^2	88.5 ± 0.0^1
Credit	84.3 ± 5.2^2	85.7 ± 5.8^1
Glass	66.8 ± 9.1^2	68.4 ± 13.7^1
Heart disease	83.0 ± 8.2^2	84.8 ± 7.9^1
Hepatitis	74.8 ± 10.6^2	77.6 ± 10.6^1
Iris	95.2 ± 5.2^2	95.5 ± 4.8^1
Soybean	81.4 ± 0.0^2	84.3 ± 0.0^1
Thyroid	92.4 ± 6.8^2	94.8 ± 6.9^1
Wine	91.1 ± 6.6^2	95.6 ± 4.8^1
Zoology	90.7 ± 5.9^1	88.8 ± 5.9^2
% Moyen	**82.1%**	**86.4%**
Class^t Moyen	**1.9**	**1.2**

TAB. 3.2 – Évaluation du système RuLe-Clust.

On observe que sur 9 bases, parmi les 10 traitées, l'organisation *a priori* des données de chaque classe en sous-classes permet d'induire des règles de meilleure qualité. En moyenne, sur ces expérimentations, l'amélioration du système pFOIL par clustering, est supérieure à 4 points[7].

Cette étude permet alors de répondre à la première question posée : organiser les classes de façon non-supervisée apporte une connaissance utile à un système de classification supervisée ; dans le cas d'une approche de classification basée sur des règles, nous observons un gain moyen supérieur à 5%, lié à la prise en compte de cette connaissance supplémentaire.

Nous avons considéré les partitions obtenues à partir des classes comme une connaissance apportée au système d'apprentissage. L'algorithme de clustering utilisé pour générer ces partitions conditionne fortement la qualité de cette connaissance. Autrement-dit, la qualité d'une partition peut, dans ce contexte, être évaluée par rapport à l'amélioration qu'elle engendre sur la performance du classifieur induit.

3.5.3 Évaluation de l'algorithme PoBOC

Nous cherchons à évaluer l'influence de l'algorithme PoBOC sur le résultat observé dans l'étude précédente. Pour cela, nous proposons de paramétrer le système RuLe-Clust avec différentes méthodes de classification non-supervisées, parmi les plus couramment utilisées.

Les méthodes choisies sont les suivantes :

[7] Cette amélioration correspond à un gain de 5% par rapport au système pFOIL.

kMED flou : Cette méthode correspond à l'algorithme k-médoïdes flou, présenté dans le chapitre 1, section 1.6.2. Une partition floue est proposée, puis transformée en une pseudo-partition, par affectation de chaque exemple à un ou plusieurs clusters, en utilisant la procédure de multi-affectations de PoBOC (cf. chapitre 2, section 2.2.2).

kMED : Il s'agit de l'algorithme k-médoïdes classique, produisant une partition stricte des exemples d'une classe.

Hiérarchique simple/moyen/complet : Par opposition aux précédentes approches par partitionnement, nous testons l'utilisation des méthodes agglomératives hiérarchiques traditionnelles du lien simple (SLINK), moyen (ALINK) et complet (CLINK) déjà présentées dans le chapitre 1, en section 1.6.1.

Les résultats de ces différents paramétrages du système RuLe-Clust sont reportées dans le tableau 3.3.

Base	RuLe-Clust (PoBOC)	RuLe-Clust (kMED flou)	RuLe-Clust (kMED)	RuLe-Clust hiérarchique (simple)	(moyen)	(complet)
Audiology	88.5±0.0[1]	84.6±0.0[2]	84.6±0.0[2]	76.9±0.0[4]	73.1±0.0[6]	76.9±0.0[4]
Credit	85.7±5.8[1]	84.9±5.2[3]	85.7±4.9[1]	83.8±4.9[4]	58.7±7.3[6]	60.6±10.3[5]
Glass	68.4±13.7[2]	57.4±12.3[3]	70.0±12.7[1]	43.2±9.4[4]	30.5±8.7[6]	42.1±11.5[5]
Heart dis.	84.8±7.9[1]	82.3±10.6[3]	83.3±8.2[2]	71.3±12.6[5]	64.2±15.1[6]	78.0±13.4[4]
Hepatitis	77.6±10.6[2]	74.5±11.2[3]	79.9±10.3[1]	71.7±10.8[4]	49.5±11.1[6]	58.5±11.8[5]
Iris	95.5±4.8[3]	95.6±4.4[2]	95.1±4.9[5]	96.0±4.4[1]	90.8±7.1[6]	95.3±4.8[4]
Soybean	84.3±0.0[1]	81.7±0.0[3]	83.3±0.0[2]	70.2±0.0[5]	71.5±0.0[4]	67.0±0.0[6]
Thyroid	94.8±6.9[2]	95.7±7.5[1]	93.3±7.1[3]	92.9±7.8[4]	56.7±13.7[6]	85.7±10.4[5]
Wine	95.6±4.8[1]	93.9±5.9[3]	94.5±5.0[2]	89.8±7.4[5]	90.3±6.3[4]	89.4±7.2[6]
Zoology	88.8±5.9[3]	88.7±6.6[4]	89.5±6.3[1]	87.5±7.6[5]	88.9±5.9[2]	86.8±7.5[6]
% Moyen	**86.4%**	**83.9%**	**85.9%**	**78.3%**	**67.4%**	**74.0%**
Clas^t Moy.	**1.7**	**2.7**	**2.0**	**4.1**	**5.2**	**5.0**

TAB. 3.3 – Comparaison de différentes méthodes de clustering.

On observe, en premier lieu, que les méthodes hiérarchiques ne sont pas adaptées pour dégager une organisation "naturelles" d'une classe d'exemples. En effet, les trois algorithmes hiérarchiques utilisés induisent les trois moins bons classifieurs de l'étude. De plus, les règles apprises à partir de cette décomposition permettent de classer correctement moins d'exemples que celles apprises dans le système de base pFOIL (jusqu'à 15 points de perte sur le taux de bonne classification).

Ce résultat peut s'expliquer par le fait que les dendrogrammes produits par les Classifications Ascendantes Hiérarchiques (CAH) ne permettent pas de dégager des partitions brutes, de qualité. Une étape d'optimisation de ces partitions pourrait améliorer sensiblement la pertinence des sous-classes proposées, et donc des règles générées.

Les algorithmes de clustering produisant les meilleures décompositions sont donc les méthodes de clustering par partitionnement. Sur 5 des 10 bases traitées, PoBOC surclasse toutes les autres approches de décomposition testées. Il se classe premier en moyenne. Cette tendance est confirmée par le taux moyen de bonne classification en utilisant PoBOC, supérieur de 0.5 points par rapport à l'algorithme k-médoïdes (strict) et de 2.5 points par rapport à la version floue.

Après cette seconde étude, situant PoBOC comme l'algorithme de clustering le mieux adapté pour dégager une organisation "naturelle" des exemples d'une classe, nous cherchons à en découvrir la/les raison(s). PoBOC est-il approprié parce que cet algorithme produit des clusters non-disjoints ? Ou est-ce simplement dû à la façon dont les clusters sont déterminés ?

3.5.4 Influence d'une pseudo-partition

Comme nous l'avons déjà fait dans le chapitre 2, nous utilisons une variante de l'algorithme PoBOC, afin de générer une partition stricte. Dans la phase de multi-affectations, plutôt que d'affecter un objet à *ses* plus proches pôles, nous choisissons de l'affecter à *son* plus proche pôle. La partition stricte $\mathcal{C}'$ obtenue par cette variante de PoBOC est, en quelque sorte, incluse dans la pseudo-partition $\mathcal{C}$ que génère l'algorithme original :

$$\forall C'_i \in \mathcal{C}',\ \exists j \text{ tel que } C'_i \subseteq C_j$$

Le tableau 3.4 rend compte de l'intérêt ou non d'organiser les exemples en sous-classes non-disjointes pour apprendre un ensemble de règles pertinentes.

Base	RuLe-Clust (PoBOC stricte)	RuLe-Clust (PoBOC)
Audiology	88.5±0.0[1]	88.5±0.0[1]
Credit	86.0±5.3[1]	85.7±5.8[2]
Glass	65.8±15.8[2]	68.4±13.7[1]
Heart disease	84.5±7.5[2]	84.8±7.9[1]
Hepatitis	77.6±10.6[1]	77.6±10.6[1]
Iris	95.3±5.0[2]	95.5±4.8[1]
Soybean	83.8±0.0[2]	84.3±0.0[1]
Thyroid	93.3±7.7[2]	94.8±6.9[1]
Wine	95.4±4.9[2]	95.6±4.8[1]
Zoology	88.8±5.9[1]	88.8±5.9[1]
% Moyen	**85.9%**	**86.4%**
Class[t] Moyen	**1.6**	**1.1**

TAB. 3.4 – Influence des intersections entre clusters.

On remarque alors que parmi les ensembles de données utilisés, seule une base semble plus adaptée à une organisation des données en sous-classes disjointes (Credit). Sur les 9 autres bases, la performance des classifieurs induits est soit identique, soit améliorée par l'algorithme PoBOC initial.

Notons également que, indirectement, la version stricte de PoBOC, permet en moyenne, de classer correctement autant d'exemples que l'algorithme k-médoïdes utilisé dans l'étude précédente. Ces deux dernières remarques nous amènent à la conclusion suivante : l'utilisation de PoBOC dans le système RuLe-Clust est appropriée, d'une part car les sous-classes découvertes correspondent aux meilleures classifications obtenues (PoBOC strict équivaut à k-médoïde), et d'autre part, grâce au fait que ces sous-classes s'intersectent (PoBOC meilleur que PoBOC stricte).

3.5.5 Comparaison avec d'autres classifieurs

Enfin, nous terminons ces expérimentations par une étude permettant de positionner le système RuLe-Clust par rapport à d'autres systèmes de référence dans le domaine de la classification supervisée. Précisons que notre objectif n'était pas de proposer une méthode plus performante que des systèmes robustes tels que C4.5 ou C5.0. Cependant, une telle étude comparative est légitimée par une certaine curiosité.

Pour cela, nous observons, toujours dans des conditions identiques d'apprentissage, le taux de bonne classification obtenu par construction d'un arbre de décision avec et sans élagage (C4.5 [149]), puis par un classifieur basé sur les instances (1 Plus Proche Voisin).

Base	RuLe-Clust (PoBOC)	C4.5 (avec élagage)	C4.5 (sans élagage)	1-PPV
Audiology	88.5±0.0[1]	84.6±0.0[2]	80.8±0.0[3]	80.8±0.0[3]
Credit	85.7±5.8[2]	86.2±5.1[1]	82.7±4.9[3]	78.5±5.0[4]
Glass	68.4±13.7[4]	72.1±10.5[3]	73.2±12.1[2]	76.8±8.6[1]
Heart disease	84.8±7.9[2]	91.7±0.0[1]	87.8±4.8[3]	83.5±9.1[4]
Hepatitis	77.6±10.6[4]	80.5±6.2[3]	81.2±7.2[1]	79.6±7.8[2]
Iris	95.5±4.8[1]	95.3±4.9[2]	95.2±5.1[3]	94.8±4.1[4]
Soybean	84.3±0.0[3]	86.7±0.0[1]	85.6±0.0[2]	74.7±0.0[4]
Thyroid	94.8±6.9[3]	94.8±7.8[3]	95.2±7.4[2]	97.6±3.8[1]
Wine	95.6±4.8[1]	93.9±6.0[3]	93.9±6.0[3]	95.5±4.7[2]
Zoology	88.8±5.9[4]	90.7±6.5[3]	91.2±6.8[2]	94.4±5.6[1]
% Moyen	**86.4%**	**87.7%**	**86.7%**	**85.6%**
Class[t] Moyen	**2.5**	**2.2**	**2.4**	**2.6**

TAB. 3.5 – Comparaison de différents classifieurs.

Les résultats, présentés dans le tableau 3.5, indiquent que RuLe-Clust, C4.5 (avec élagage) et 1-PPV, obtiennent chacun les meilleurs taux de bonne classification sur trois des dix bases de test et se placent second sur deux autres bases. En moyenne, la performance du système RuLe-Clust est comparable à la méthode C4.5 sans élagage. Ce résultat est meilleur d'environ 1 point, à celui de la méthode basée sur les instances (1-PPV), et inférieur dans les mêmes proportions, à la performance de C4.5 avec élagage.

L'élagage d'un arbre de décision consiste à simplifier l'arbre par la suppression et la fusion de certaines branches, dans le but de limiter la sur-adéquation avec les données d'entraînement. Ce traitement, généralement postérieur à la phase d'apprentissage, améliore sensiblement les performances du système, comme nous le constatons dans notre expérimentation. On peut alors supposer qu'un traitement similaire améliorerait également les performances du système RuLe-Clust. Deux stratégies (éventuellement complémentaires) sont envisageables : d'une part l'intégration de la notion de "tolérance", en autorisant chaque règle à couvrir une proportion raisonnable d'exemples négatifs ; d'autre part la simplification *a posteriori* de l'ensemble de règles générées.

3.6 Conclusion

Dans ce chapitre, nous avons présenté le système RuLe-Clust (*Rule-Learning from Clustering*) permettant d'apprendre un ensemble de règles de classification, à partir d'une décomposition des données en sous-classes "naturelles". Deux objectifs étaient alors visés : d'une part, nous cherchions à adapter ce principe de décomposition, déjà utilisé dans des systèmes d'apprentissage basé sur les instances, au cas de la classification par apprentissage de règles ; d'autre part, dans la continuité des deux chapitres précédents, le système RuLe-Clust nous a semblé offrir un cadre applicatif intéressant, pour une évaluation objective de l'algorithme PoBOC.

Nous avons alors motivé le recours à un processus de classification non-supervisée, afin de cibler des sous-classes "naturelles", susceptibles d'induire chacune une règle pertinente. Ce principe général de décomposition a été présenté comme alternative aux systèmes d'apprentissage séquentiels : AQ, CN2 ou encore pFOIL.

Dans un deuxième temps, nous nous sommes intéressé à l'influence que peuvent avoir les recouvrements entre clusters, issus du processus de décomposition, sur la qualité prédictive des règles induites. Nous avons alors montré que, dans certaines situations, ce type de schémas peut aider à orienter la construction des règles vers une couverture plus "naturelle" des classes.

Enfin, les expérimentations proposées ont permi de confirmer les hypothèses formulées. Sur dix ensembles de tests, traditionnellement utilisés pour évaluer les systèmes de classification supervisée, nous avons observé d'abord que RuLe-Clust génère un ensemble de règles davantage pertinentes que le système équivalent procédant par couverture séquentielle. L'utilisation de la connaissance induite par le processus de clustering permet de "hisser" un système du type pFOIL, à un niveau de performance approchant les meilleurs systèmes existants (*e.g.* C4.5).

Finalement, l'algorithme PoBOC a été évalué avec succès, puisque nous avons montré empiriquement, que la décomposition proposée par cet algorithme se traduit (*via* la génération d'un ensemble de règles), par un taux de bonne classification supérieur aux algorithmes de clustering traditionnels (hiérarchiques ou par partitionnement). Cette supériorité étant à attribuer, d'une part, à la qualité générale des clusters obtenus, et d'autre part, aux intersections entre ces clusters.

4

Apprentissage disjonctif par décomposition de concepts

Sommaire

4.1 Introduction . . . 104
4.2 Bases de la logique du premier ordre . . . 104
4.3 Apprentissage disjonctif . . . 105
4.3.1 Le système FOIL . . . 106
4.3.2 Le système GOLEM . . . 107
4.4 Extension du système RuLe-Clust à la logique du premier ordre 108
4.4.1 Apprentissage disjonctif d'un concept . . . 108
4.4.2 L'approche par décomposition . . . 109
4.4.3 Exemple 1 : sous-concepts non-disjoints I . . . 110
4.4.4 Exemple 2 : description numérique . . . 111
4.5 Mesure de similarité . . . 113
4.5.1 Langage infini . . . 114
4.5.2 Expérimentations . . . 117
4.6 Vue générale de la méthode . . . 122
4.6.1 Apprentissage d'une clause . . . 123
4.6.2 Complexité de la méthode globale . . . 123
4.6.3 Expérimentations . . . 124
4.7 Conclusion . . . 126

4.1 Introduction

Dans ce quatrième chapitre, nous proposons d'étendre le principe général illustré par le système RuLe-Clust, à l'apprentissage de règles en logique du premier ordre (avec variables). Comme nous allons le voir, ce formalisme est plus expressif et permet de traiter des problèmes difficiles voire impossibles à représenter dans le formalisme propositionnel, utilisé jusqu'ici. Cependant, cette transition n'est pas sans poser des problèmes, récurrents en Programmation Logique Inductive (ILP), comme par exemple, la difficulté à tester la couverture d'un exemple par une règle ou la construction même de cette règle. L'approche par décomposition, que nous avons étudié dans le chapitre précédent, implique également la définition d'une mesure de similarité, qui semble moins évidente en premier ordre.

Après quelques rappels importants concernant la théorie de la logique du premier ordre, le problème général de l'apprentissage disjonctif dans ce formalisme est énoncé. Nous présentons alors deux stratégies de base pour la construction d'une disjonction de règles logiques, illustrées par les systèmes FOIL [150] et GOLEM [132]. Le principe d'apprentissage disjonctif par décomposition de concepts est ensuite introduit, et complété par la proposition d'une mesure de similarité sur des objets décrits dans ce formalisme. Des exemples adaptés, permettent tout au long de cette étude d'illustrer et d'évaluer les deux principaux choix opérés pour cette approche : la mesure de similarité et l'algorithme de classification non-supervisée PoBOC.

4.2 Bases de la logique du premier ordre

Nous considèrons le problème qui consiste à apprendre le concept *grand-père*, à partir de la base de connaissances suivante :

mère(françois, marie), père(marie, jean), père(paul, pierre),
père(pierre, jean), père(jeanne, pierre)

Dans cette base de connaissance (BK[1]), l'élément *mère(françois, marie)* doit être interprété de la manière suivante : *"françois a pour mère marie"*. Ce concept peut être défini par les deux règles

grand-père(X,Y)←père(X,Z),père(Z,Y),
grand-père(X,Y)←mère(X,Z),père(Z,Y).

Dans cet exemple, *grand-père, mère* et *père* sont les symboles de prédicat (tous d'arité 2), *marie, françois, pierre, jean, paul* et *jeanne* les constantes, X, Y et Z sont des variables.

- Un *littéral* est un atome (ou la négation d'un atome), de la forme $p(t_1, \ldots, t_n)$ où p est un prédicat et où les t_i représentent des variables ou constantes[2].
- Les deux règles, présentées ci-dessus, sont appelées des *clauses* (ou règles logiques), telles que la partie gauche désigne la *tête* (ou le *conséquent*) tandis que la partie droite désigne le *corps* de la clause (ou les *antécédents*). Le corps d'une clause est constitué par la conjonction de littéraux (positifs) et la tête d'un unique littéral.

[1] *Background Knowledge.*

[2] Précisons que l'on se restreint, dans cette étude, à des langages en logique du premier ordre, ne contenant pas de symboles de fonction.

– Un *atome clos* est un atome ne contenant aucune variable (e.g. *père(pierre, jean)*).

Définition 4.1. *Soient deux ensembles d'atomes clos E^+ et E^- et une base de connaissances BK. Un exemple e est couvert par une clause $C : p(X) \leftarrow l_1, \ldots l_n$ relativement à BK si et seulement si il existe une instanciation σ des variables de C telle que $l_1\sigma, \ldots l_n\sigma$ est vrai dans $BK \cup E^+$ et $p(X)\sigma = e$.*

Dans notre exemple, les atomes clos *grand-père(françois, jean), grand-père(paul, jean)* et *grand-père(jeanne, jean)* correspondent aux exemples positifs (E^+) du concept cible *grand-père*. L'exemple *grand-père(françois, jean)* est couvert par la première clause *via* l'instanciation σ_1 ={*X/françois, Z/marie, Y/jean*} ; les exemples *grand-père(paul, jean)* et *grand-père(jeanne, jean)* sont couverts par la seconde clause *via* les instanciations respectives suivantes : σ_2 ={*X/paul, Z/pierre, Y/jean*} et σ_3 ={*X/jeanne, Z/pierre, Y/jean*}.

Le formalisme logique présenté ci-dessus, permet une représentation plus riche, que le formalisme propositionnel. Les clauses logiques permettent également une définition récursive d'un concept, ce qui est impossible dans le formalisme propositionnel. Par exemple, si on souhaite apprendre le concept *ancêtre*, une définition récursive naturelle est :

ancêtre(X, Y)←père(X, Y)
ancêtre(X, Y)←mère(X, Y)
ancêtre(X, Y)←père(X,Z),ancêtre(Z, Y)
ancêtre(X, Y)←mère(X,Z),ancêtre(Z, Y).

Dans une telle définition récursive, le prédicat de tête est utilisé également dans le corps de la clause.

4.3 Apprentissage disjonctif

Un concept est généralement appris par construction d'un ensemble de clauses, dont la disjonction permet de couvrir l'ensemble des instances positives pour ce concept cible. Il existe une définition triviale de ce concept, permettant de couvrir tous les exemples positifs et de rejeter tous les exemples négatifs (excepté lorsqu'un exemple possède à la fois les deux étiquettes positif et négatif), cette définition correspond à la disjonction des exemples positifs. Or, comme nous l'avons déjà évoqué dans le cadre propositionnel, ce résultat n'est pas (ou peu) réutilisable pour le traitement d'une nouvelle instance. De plus, l'espace des hypothèses est très important si l'on considère les disjonctions. Il devient alors impossible de l'explorer totalement.

Les systèmes proposés pour apprendre un ensemble de clauses caractéristiques[3] d'un concept cible, doivent tenir compte de cet aspect. Ils s'accompagnent donc généralement d'une limitation de l'espace des hypothèses par un biais syntaxique sur les clauses (taille et forme des clauses). Ces systèmes peuvent être vus, pour les plus connus, comme des extensions des systèmes d'apprentissage de règles propositionnelles (chapitre 3). Par exemple, le système FOIL [150] constitue une extension des systèmes utilisant un algorithme de couverture séquentielle, tels que CN2 ou pFOIL.

[3]Dans la suite, nous parlerons d'une clause "caractéristique" d'un ensemble d'exemples ou d'un concept, si cette clause couvre l'ensemble de ces exemples et rejette tous les exemples négatifs de ce concept. Il ne s'agit en aucun cas d'une "règle de caractérisation", utilisées dans d'autres perspectives en Fouille De Données.

4.3.1 Le système FOIL

FOIL [150] recherche une définition disjonctive d'un concept par construction itérative d'une description conjonctive qui couvre des exemples positifs et rejette les exemples négatifs. Nous présentons l'algorithme général en figure 4.1.

Algorithme FOIL :

Soit Pos l'ensemble des exemples pour lesquels le prédicat cible est $Vrai$,
Soit Neg l'ensemble des exemples pour lesquels le prédicat cible est $Faux$,
Soit R, l'ensemble des règles apprises, initialisé à vide.

Tant que Pos n'est pas vide : *(Apprendre une règle)*

r est une règle contenant le prédicat cible en tête et de corps vide
Soit $Neg' = Neg$
Tant que Neg' n'est pas vide : *(Spécialiser r par ajout d'un littéral)*

Soit C l'ensemble des littéraux candidats pour la règle r
Sélectionner $l \in C$ tel que l maximise une mesure de gain
Ajouter l au corps de r
Supprimer de Neg', l'ensemble des exemples rejetés par r

Ajouter r à l'ensemble des règles apprises R
Supprimer de Pos, l'ensemble des exemples couverts par r

Retourner R.

FIG. 4.1 – L'algorithme d'apprentissage de règles logiques FOIL.

Chaque clause dans FOIL est générée par une approche du plus général au plus spécifique. Partant de la clause la plus générale (e.g. *ancêtre(X, Y)*←), FOIL spécialise la clause par construction de l'ensemble des littéraux candidats, puis sélection du "meilleur" littéral parmi cet ensemble.

L'ensemble des littéraux candidats est défini par tous les littéraux qui vérifient l'une des formes suivantes :

- $p(V_1, \ldots, V_n)$ ou $\neg p(V_1, \ldots, V_n)$, où p est un symbole de prédicat et l'une au moins des variables $V_1, \ldots, V_n$ apparaît déjà dans la règle en cours de construction[4],
- $egal(V_i, V_j)$ ou $\neg egal(V_i, V_j)$, où V_i et V_j apparaissent déjà dans la règle en cours de construction.

Parmi les symboles de prédicats p, le prédicat associé au concept cible (prédicat de la tête) peut être considéré, de façon à générer une définition récursive (pour définir *ancêtre(X, Y)* par exemple), avec certaines restrictions sur les variables.

La sélection du meilleur littéral parmi l'ensemble des littéraux candidats est effectuée

[4]Le symbole "¬" correspond à la négation. FOIL propose en effet un langage plus expressif que les clauses de Horn, en autorisant les littéraux négatifs.

au moyen d'une mesure de gain. Cette mesure évalue les proportions d'exemples positifs et négatifs couverts par la clause, avant et après l'ajout du littéral évalué.

Il est très fréquent que l'heuristique "gloutonne" utilisée dans FOIL n'aboutisse pas à la construction d'une clause (de complexité raisonnable) rejetant tous les exemples négatifs du concept cible. Ainsi, un processus de retour arrière (*backtracking*) est généralement envisagé, pour reconsidérer le dernier littéral ajouté en le remplaçant par un autre littéral un peu moins discriminant. Dans FOIL, l'espace de recherche est alors borné, d'une part en limitant la complexité des clauses possibles à générer, d'autre part en ne considérant à chaque ajout, que les littéraux les plus discriminants.

4.3.2 Le système GOLEM

Contrairement à FOIL, le système GOLEM [132] utilise une stratégie de construction de règle du plus spécifique vers le plus général. Cet algorithme est présenté dans la figure 4.2

Algorithme GOLEM :

Soit Pos l'ensemble des exemples positifs non couverts,
Soit Neg l'ensemble des exemples négatifs,

Soit E un ensemble de k paires d'exemples $< e_i, e_j >$ de Pos (choisies aléatoirement)
Soit C, l'ensemble de tous les PPG[a] : $C = \{PPG(e_i, e_j) | < e_i, e_j > \in E\}$,
Supprimer de C les hypothèses qui couvrent des exemples négatifs
Sélectionner dans C l'hypothèse g qui couvre le plus d'exemples de Pos
Supprimer de Pos les exemples couverts par g
Tant que g augmente sa couverture sur Pos :

> **Soit** E' un ensemble de k exemples de Pos (choisis aléatoirement),
> **Soit** C, l'ensemble de tous les PPG : $C = \{PPG(g, e') | e' \in E'\}$,
> Supprimer de C les hypothèses qui couvrent des exemples négatifs,
> Sélectionner dans C l'hypothèse g qui couvre le plus d'exemples positifs,
> Supprimer de Pos les exemples couverts par g,

Retourner : l'hypothèse g.

[a] Plus Petit Généralisé.

FIG. 4.2 – Algorithme de construction d'une règle par généralisation : GOLEM.

Chaque clause générée par GOLEM est construite à partir d'un tirage de k paires d'exemples positifs (généralement $k = 2$), et pour chaque paire, une hypothèse spécifique pertinente est recherchée, puis généralisée. La construction d'une clause basée sur le tirage d'un ou plusieurs exemples se retrouve dans les systèmes INDUCE [84] ou encore PROGOL [131] - ce dernier utilisant une approche de construction par spécialisation. Le problème du choix des exemples est récurrent pour ce type d'approche. Dans GOLEM, ce problème est

en partie résolu, par le tirage de plusieurs exemples, donnant lieu à différentes hypothèses parmi lesquelles seulement la meilleure est retenue pour être généralisée.

Notons que le système GOLEM utilise la notion de *Plus Petit Généralisé* (PPG) [144]. On peut parler du PPG de termes, de littéraux ou de clauses. Étant données des descriptions spécifiques d'un concept, il s'agit de construire une définition de ce concept, plus générale que les descriptions des exemples, et la moins générale possible parmi celles généralisant les exemples.

Enfin, la meilleure hypothèse correspond à celle qui couvre le plus d'exemples positifs, parmi ceux non-encore couverts.

Nous avons présenté deux systèmes d'apprentissage d'une disjonction de clauses permettant de caractériser un concept. Ces deux systèmes sont représentatifs des deux approches de construction de clauses : du plus général au plus spécifique (FOIL) et du plus spécifique au plus général (GOLEM). Nous choisissons, dans la suite, de conserver la stratégie utilisée dans le système RuLe-Clust (cf. chapitre 3), à savoir la couverture d'un ensemble d'exemples par spécialisation.

4.4 Extension du système RuLe-Clust à la logique du premier ordre

4.4.1 Apprentissage disjonctif d'un concept

L'introduction de la disjonction dans l'espace des hypothèses est importante parce que beaucoup de concepts peuvent être vus de façon disjonctive et leur définition nécessite alors plusieurs règles. Par exemple, le concept *grand-père*, étant donné le langage que nous avons défini, se compose en fait de deux sous-concepts : *"grand-père paternel"* et *"grand-père maternel"*. Ce concept peut donc être défini par deux clauses :

grand-père(X,Y)←père(X,Z),père(Z,Y).
grand-père(X,Y)←mère(X,Z),père(Z,Y).

chaque règle définissant un sous-concept du concept initial. Ainsi, les instances sont divisées naturellement en deux sous-ensembles d'exemples (ou groupes), satisfaisant à l'une ou l'autre des deux règles. Le principe du système d'apprentissage que nous proposons ici, consiste à rechercher d'abord des groupes d'exemples, afin de guider, ensuite, la génération des clauses.

Nous motivons cette approche par le fait que les systèmes tels que FOIL, utilisant une approche gloutonne pour construire les clauses, dépendent essentiellement de fonctions heuristiques qui induisent un biais très important dans la recherche des littéraux à ajouter.

Les fonctions heuristiques utilisées, comme par exemple la mesure d'*Utilité* (utilisée pour FOIL) permettent de mesurer le pouvoir discriminant (noté $\Gamma(l, C)$) d'un littéral l par rapport à une clause C. Soient E^+ (resp. E^-) l'ensemble des exemples positifs (resp. négatifs) et $cov(E, C)$ le nombre d'exemples[5] de E couverts par la clause C, $\Gamma(l, C)$ dépend au minimum des deux valeurs suivantes : $cov(E^+, C \cup l)$ et $cov(E^-, C \cup l)$ où $C \cup l$ désigne la clause obtenue par l'ajout du littéral l dans le corps de C. La définition de $\Gamma(l, C)$ joue un rôle central dans le processus, et donc sur l'ensemble des sous-concepts induits. De plus,

[5]En Programmation Logique Inductive, le nombre d'instanciations de C couvrant un élément de E peut également être utilisé pour définir le pouvoir discriminant.

$\Gamma(l, C)$ est fortement dépendant des ensembles E^+ et E^- : quelques changements dans ces ensembles peuvent conduire à des solutions très différentes, et donc à des sous-concepts différents.

Enfin, si $h \leftarrow p, q$ est une clause couvrant certains exemples positifs et aucun exemple négatif, les valeurs $\Gamma(p, h \leftarrow)$ et $\Gamma(q, h \leftarrow)$ ne sont pas nécessairement élevées. Par conséquent, il peut être impossible, pour une méthode gloutonne, d'apprendre la clause $h \leftarrow p, q$ parce que, partant de la clause générale $h \leftarrow$, ni p, ni q ne semblent être discriminants. Ce problème est plus général que celui des littéraux déterminés, étudié dans [150, 163].

4.4.2 L'approche par décomposition

Le principal inconvénient des méthodes gloutonnes est que chaque concept est caractérisé par une clause $h \leftarrow b_1, \ldots, b_n$ où b_i est un littéral fortement discriminant pour la clause $h \leftarrow b_1, \ldots, b_{i-1}$. Plutôt que de générer des sous-concepts dépendant d'une fonction de gain, le système RuLe-Clust propose un processus en trois étapes :

(1) définir une mesure de similarité sur l'ensemble des exemples,
(2) décomposer chaque concept en sous-concepts d'exemples similaires,
(3) pour chaque sous-concept, construire une unique règle caractéristique.

Le principal avantage de cette approche est de réduire fortement l'espace de recherche pour l'étape de construction d'une définition pour un sous-concept : seules les hypothèse couvrant tous les exemples positifs pour ce sous-concept sont considérées[6]. En pratique, soit G un sous-concept (ensemble d'exemples positifs) et supposons que $C = h \leftarrow b_1, \ldots, b_i$ est la clause en construction. Un littéral l peut être ajouté au corps de C seulement si $cov(G, C \cup l) = |G|$. Ainsi, le pouvoir discriminant ne correspond plus à la combinaison de deux critères ($cov(E^+, C \cup l)$ et $cov(E^-, C \cup l)$) mais est principalement déterminé par $cov(E^-, C \cup l)$. Par exemple, on peut choisir d'ajouter à C le littéral l qui minimise $cov(E^-, C \cup l)$ parmi l'ensemble des littéraux l_i tels que $cov(G, C \cup l_i) = |G|$.

Conformément à l'algorithme de décomposition utilisé dans le système RuLe-Clust, lorsqu'il est impossible de trouver une clause qui définit le sous-concept G, alors G est re-décomposé en de nouveaux sous-concepts pour chacun desquels on recherchera une définition. Ce processus est répété jusqu'à ce qu'une clause puisse être apprise pour chaque sous-concept. Dans le pire des cas, cela conduit à des sous-concepts contenant un seul exemple positif, entraînant alors une situation d'*over-fitting*, que nous n'avons jamais rencontré lors de nos expérimentations. Si aucune clause ne peut être trouvée pour un tel sous-concept, cela signifie qu'il n'y a pas de solution complète[7] et consistante[8] pour le problème initial d'apprentissage.

Dans les algorithmes gloutons, la définition de $\Gamma(l, C)$ joue un rôle central alors que dans notre approche, c'est la décomposition des concepts en sous-concepts qui est essentielle. Un point important concerne alors l'évaluation de notre méthode de clustering dans ce contexte, *i.e.* tester dans quelle mesure la décomposition effectuée conduit à de "bons" sous-concepts. La qualité d'une décomposition peut être considérée selon différents points de vue non exclusifs :

[6] Cette contrainte peut être allégée afin de tenir compte de la présence éventuelle de données bruitées : seules les hypothèses couvrant une proportion importante du sous-concept seront alors considérées.

[7] Une solution complète est une hypothèse couvrant tous les exemples positifs.

[8] Une solution consistante est une hypothèse rejetant tous les exemples négatifs.

L'auto organisation : Dans certains cas, un concept peut être naturellement divisé en sous concepts : chaque sous-concept étant constitué d'exemples similaires (qui partagent de nombreuses propriétés caractéristiques).

La simplicité : D'un point de vue pratique, une décomposition est bonne si elle nécessite peu de règles simples (courtes) pour la définir.

La précision : Enfin, la qualité des règles reflète la pertinence de la décomposition et peut être évaluée par rapport à leur pouvoir de prédiction sur de nouveaux exemples.

Rappelons que la décomposition d'un concept est un processus en deux étapes. D'abord nous définissons une mesure de similarité et calculons la similarité entre chaque paire d'exemples positifs. Ensuite, nous utilisons un algorithme de clustering afin de construire des groupes éventuellement non-disjoints, d'exemples similaires.

Afin de motiver cette approche et dans un souci de simplicité, nous proposons dans la prochaine section deux exemples qui pourraient aussi être exprimés dans un formalisme attribut/valeur. Cependant, comme nous le montrerons en section 4.5.2, notre approche peut être appliquée sur des problèmes spécifiques en ILP.

4.4.3 Exemple 1 : sous-concepts non-disjoints I

On considère la base de connaissance composée des atomes clos suivants :

$$\{r(a), r(b), r(f), r(h), s(a), s(b), s(g), s(i), t(c), t(d), t(g), t(i),$$
$$u(c), u(d), u(f), u(h), v(b), v(c), v(f),\ r(e), s(e), t(e), u(e), v(e)\}$$

On suppose que, pour un concept cible p, l'ensemble des exemples positifs est $\{p(a), p(b), p(c), p(d)\}$ et que l'ensemble des exemples négatifs est $\{p(f), p(g), p(h), p(i)\}$.

Cet exemple permet d'illustrer les limites des algorithmes gloutons, pour la construction des clauses. En effet, avec un algorithme glouton tel que FOIL, la construction débute avec la clause générale $p(X) \leftarrow$,

- si on ajoute l'un des littéraux $r(X), s(X), t(X)$ ou $u(X)$, on obtient une clause couvrant 2 exemples positifs et 2 négatifs ;
- si on ajoute le littéral $v(X)$, on obtient une clause qui couvre 2 exemples positifs et 1 négatif.

Le littéral $v(X)$ est donc choisi et la définition finale apprise contient au moins 3 clauses. Pourtant, le concept peut être caractérisé par les deux clauses suivantes :

$C_1 : p(X) \leftarrow r(X), s(X).$
$C_2 : p(X) \leftarrow t(X), u(X).$

conduisant ainsi aux deux sous-concepts $\{p(a), p(b)\}$ et $\{p(c), p(d)\}$. Pour apprendre ces deux clauses, notre approche par décomposition devrait construire deux sous-concepts à partir de la base de connaissance, *i.e.* considérer que $p(a)$ et $p(b)$ (resp. $p(c)$ et $p(d)$) sont fortement similaires.

On propose d'utiliser la mesure de similarité définie dans [120] et dont le principe général a été présenté dans le chapitre 1. Pour cette application on considère le langage $\mathcal{L}$ défini par l'ensemble suivant de propriétés, correspondant à toutes les clauses n'ayant qu'un atome dans le corps :

$$\{p(X) \leftarrow r(X),\ p(X) \leftarrow s(X),\ p(X) \leftarrow t(X),\ p(X) \leftarrow u(X),\ p(X) \leftarrow v(X)\}.$$

La similarité entre deux exemples e_i et e_j est alors définie par le nombre de clauses C dans $\mathcal{L}$ telles que e_i et e_j sont tous deux couverts par C ou e_i et e_j sont tous deux non-couverts par C. Par exemple, $p(a)$ et $p(b)$ sont couverts tous deux par les propriétés $p(X) \leftarrow r(X)$ et $p(X) \leftarrow s(X)$ et tous les deux non-couverts par $p(X) \leftarrow t(X)$ et $p(X) \leftarrow u(X)$. La similarité entre $p(a)$ et $p(b)$ est donc égale à 4. De la même façon, on calcule les similarités pour chaque paire d'exemples positifs et obtenons la matrice suivante :

	$p(a)$	$p(b)$	$p(c)$	$p(d)$
$p(a)$	5	4	0	1
$p(b)$	4	5	1	0
$p(c)$	0	1	5	4
$p(d)$	1	0	4	5

Dans cette matrice, les deux sous-concepts attendus apparaissent clairement. Cependant, dans beaucoup de problèmes, les sous-concepts ne correspondent pas à des ensembles disjoints. Par exemple, si l'on ajoute maintenant $p(e)$ comme nouvel exemple positif, la matrice de similarité devient alors :

	$p(a)$	$p(b)$	$p(e)$	$p(c)$	$p(d)$
$p(a)$	5	4	2	0	1
$p(b)$	4	5	3	1	0
$p(e)$	2	3	5	3	2
$p(c)$	0	1	3	5	4
$p(d)$	1	0	2	4	5

Dans cette matrice, les deux sous-concepts initiaux $\{p(a), p(b)\}$ et $\{p(c), p(d)\}$ apparaissent toujours, mais $p(e)$ peut être inséré dans les deux sous-concepts. En fait, le nouvel exemple est couvert par les clauses C_1 et C_2 qui induisent respectivement les deux sous-concepts $\{p(a), p(b), p(e)\}$ et $\{p(e), p(c), p(d)\}$.

L'utilisation d'une méthode de regroupement autorisant les recouvrements semble donc particulièrement adaptée, notamment dans cet exemple. Nous verrons par la suite que l'algorithme PoBOC permet d'aboutir aux deux sous-concepts attendus ici.

4.4.4 Exemple 2 : description numérique

Considérons à présent un exemple où les observations sont décrites par des propriétés numériques.

Soit un ensemble d'objets $\{a, b, \ldots, u\}$ défini sur deux attributs x et y. Ces objets sont représentés en figure 4.3 et sont exprimés dans une base de connaissances où sont définis les prédicats x, y, $\geq$ et $\leq$ (e.g. $x(U, V)$ est vrai quand l'objet U satisfait $x = V$). La base de connaissance contient les atomes clos suivants : $x(a, 1),\ y(a, 1),\ x(b, 1),\ y(b, 2), \ldots, \geq (1, 1),\ \geq (1, 2), \ldots$. Finalement, l'ensemble des exemples positifs pour le concept cible p est $\{p(a), p(b), \ldots, p(j)\}$ et l'ensemble des exemples négatifs est $\{p(l), p(m), \ldots, p(u)\}$.

Si on pose le biais syntaxique interdisant l'apparition des constantes $1, \ldots, 6$ dans les clauses apprises, il sera alors difficile, pour un algorithme glouton, d'apprendre une définition du concept cible.

En effet, à partir de la clause générale $p(X) \leftarrow$, les littéraux $x(X, Y)$ et $y(X, Y)$ sont des littéraux déterminés[9]. Aucun des deux n'est discriminant pour le concept p. Cependant

[9]Les littéraux déterminés sont des littéraux qui couvrent tous les exemples positifs et tous les exemples négatifs. Leur gain est alors très faible bien qu'ils puissent jouer un rôle important dans la construction d'une bonne définition en permettant l'introduction de nouvelles variables dans la clause.

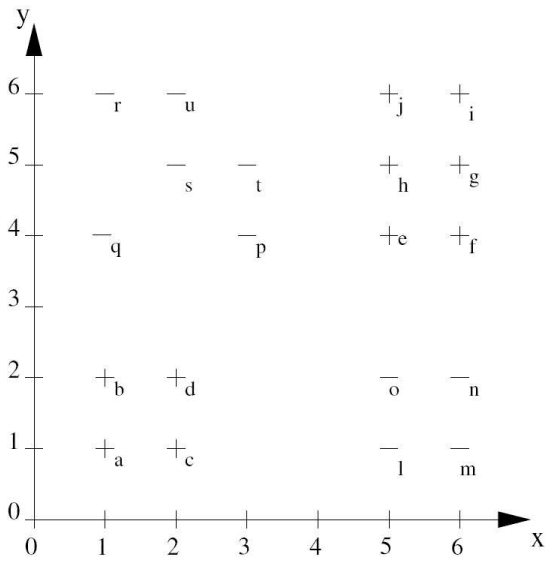


FIG. 4.3 – Exemples décrits sur deux attributs numériques.

il est généralement choisi d'ajouter tous les littéraux déterminés, de plus il s'agit ici des seuls littéraux possibles à ajouter[10].

Ainsi, à partir de la clause $p(X) \leftarrow x(X, Y_1), y(X, Y_2)$ aucun des littéraux suivant n'est discriminant :

$$\geq (Y_1, 1), \ldots, \geq (Y_1, 6), \geq (Y_2, 1), \ldots, \geq (Y_2, 6)$$

$$\leq (Y_1, 1), \ldots, \leq (Y_1, 6), \leq (Y_2, 1), \ldots, \leq (Y_2, 6)$$

Un choix arbitraire devra alors être fait, conduisant à une définition probablement différente de celle attendue.

Dans notre approche, on peut choisir de calculer la mesure de similarité à l'aide du langage constitué des 12 clauses suivantes, correspondant à toutes les clauses possibles avec un seul atome dans le corps[11] :

$$\begin{array}{ll} \{p(X) \leftarrow x(X,Y), \geq (Y,1) & p(X) \leftarrow y(X,Y), \geq (Y,1) \\ p(X) \leftarrow x(X,Y), \geq (Y,2) & p(X) \leftarrow y(X,Y), \geq (Y,2) \\ \ldots & \ldots \\ p(X) \leftarrow x(X,Y), \geq (Y,6) & p(X) \leftarrow y(X,Y), \geq (Y,6)\} \end{array}$$

La mesure de similarité définie par ce langage, est proche d'une distance euclidienne. Cette mesure permet d'induire 2 clusters correspondant aux sous-concepts

- $\{p(a), p(b), p(c), p(d)\}$ (couvert par la clause $p(X) \leftarrow x(X, Y), \leq (Y, 2), y(X, Z), \leq (Z, 2)$),
- $\{p(e), p(f), \ldots, p(j)\}$ (couvert par la clause $p(X) \leftarrow x(X, Y), \geq (Y, 4), y(X, Z), \geq (Z, 3)$).

Notons que sur les deux exemples précédents, les clauses qui caractérisent les clusters obtenus ne se basent pas sur le langage utilisé pour la mesure de similarité. Les expérimentations

[10] En tenant compte d'un biais syntaxique associé au domaine des variables X et Y.

[11] Nous étudierons, dans la suite de ce chapitre, l'influence du langage sur la mesure de similairté.

sur ces deux exemples seront détaillées dans la suite.

Une fois la mesure de similarité établie, l'algorithme de décomposition, présenté dans le chapitre précédent, peut être utilisé sans aucune modification. La première difficulté, liée à l'adaptation du système RuLe-Clust dans un formalisme logique, concerne donc la définition de cette mesure de similarité.

4.5 Mesure de similarité

Plusieurs approches ont été proposées afin de définir une mesure de similarité entre objets décrits en logique du premier ordre. La méthode habituelle pour construire une mesure de similarité consiste d'abord à produire une description des objets. Ensuite, pour des descriptions basées sur des ensembles de littéraux, la similarité est calculée par l'intersection de ces descriptions [14, 52] ; pour des descriptions basées sur des règles, la similarité entre deux objets est donnée par le nombre de règles satisfaites par les deux objets [163, 162, 120]. Dans tous les cas, deux objets sont considérés comme similaires lorsqu'ils partagent des propriétés. Dans [120], la similarité est définie par rapport à un ensemble fini de clauses, appelé langage ; pour une clause C et un exemple e, on définit la fonction $couvert(C, e)$ comme suit :

$$couvert(C, e) = \begin{cases} 1 & \text{si } e \text{ est couvert par } C \\ 0 & \text{sinon.} \end{cases}$$

Étant donné un langage $\mathcal{L}$, on définit $\mathcal{L}(e_i, e_j)$ comme l'ensemble des clauses C telles que e_i et e_j sont, soit tous les deux couverts par C, soit tous les deux rejetés par C :

$$\mathcal{L}(e_i, e_j) = \{C \in \mathcal{L} | couvert(C, e_i) = couvert(C, e_j)\}$$

La similarité entre e_i et e_j, notée $sim_{\mathcal{L}}(e_i, e_j)$, est alors définie par :

$$sim_{\mathcal{L}}(e_i, e_j) = \frac{|\mathcal{L}(e_i, e_j)|}{|\mathcal{L}|}$$

Cette mesure de similarité est celle que nous avons utilisé dans l'exemple 1 (section 4.4.3). Cependant, cette mesure ne tient pas compte de l'ensemble des exemples négatifs. Nous proposons d'étendre cette mesure en donnant un poids aux clauses du langage, en utilisant cette fois l'ensemble des exemples positifs et négatifs comme indicateur de pertinence. Plus précisément, si E^+ et E^- désignent respectivement les ensembles d'exemples positifs et négatifs, le poids d'une clause C est défini par :

$$w(C) = \begin{cases} \frac{cov(E^+, C)}{cov(E^+, C) + cov(E^-, C)} & \text{si } cov(E^+, C) > 0 \\ 0 \text{ sinon.} \end{cases}$$

La valeur maximum de ce poids est égale à 1, pour les clauses qui ne couvrent que des exemples positifs. La mesure de similarité pondérée que nous obtenons est donc :

$$w\_sim(e_i, e_j) = \frac{\sum_{C \in \mathcal{L}(e_i, e_j)} w(C)}{|\mathcal{L}|}$$

Cette pondération permet de donner plus d'importance aux clauses plus "proches" du concept cible. La mesure peut cependant présenter un inconvénient lorsque le langage

contient un nombre important de clauses avec un faible poids, puisqu'il y a un risque de réduire l'influence des clauses de poids fort. Pour cette raison, nous proposons une définition de la similarité, relativement à un seuil α, en dessous duquel les clauses ne sont plus considérées :

$$w\_sim_\alpha(e_i, e_j) = \frac{\sum_{C \in \mathcal{L}(e_i,e_j), w(C)>\alpha} w(C)}{|\mathcal{L}|}$$

La clause la plus générale, couvrant tous les exemples positifs et tous les exemples négatifs, a un poids égal à $|E^+|/(|E^+|+|E^-|)$. Toutes les clauses ayant un poids inférieur à cette valeur ont une précision[12] inférieure à la clause la plus générale. Pour cette raison, en pratique on utilise la mesure de similarité w_sim_α avec le paramètre $\alpha = |E^+|/(|E^+|+|E^-|)$.

4.5.1 Langage infini

Dans certains cas, on peut être amené à considérer un langage infini, ou trop grand pour être complètement évalué sur tous les exemples. Dans cette situation, on considère un échantillon fini du langage : les clauses sont générées stochastiquement à l'aide de paramètres, spécifiés à l'avance, et permettant de contrôler la taille du langage ($|\mathcal{L}|$), la complexité des clauses ($P_{lit.}$, $P_{contr.}$) et les poids des domaines (P_{dom}).

Pour mieux comprendre l'utilité de ces paramètres, nous construisons un exemple ad-hoc. On considère dans ce qui suit, un langage constitué de trois prédicats p, q et r, d'arités respectives 1,3 et 3. Pour les arguments de chaque prédicat, on pose les domaines suivants :

$$p(D_1) \;\; ; \;\; q(D_1, D_2, D_3) \;\; ; \;\; r(D_2, D_4, D_5)$$

Ainsi, les premiers arguments des prédicats, respectivement p et q, sont de même domaine, tout comme le second argument de q et le premier de r.

Supposons que $p(X)$ représente le concept cible à définir. Le langage de description que nous allons construire est alors constitué de clauses de la forme :

$$p(X) \leftarrow l_1, l_2, \ldots, l_n, c_1, \ldots, c_m$$

Dans cette clause, les littéraux l_i sont de la forme $q(X, Y, Z)$ ou $r(Y, T, U)$ et les c_j correspondent à des contraintes du type $T = v$, où v est une valeur choisie dans le domaine de T[13].

Spécification de la taille des clauses du langage

Le paramètre $P_{lit.}$ permet de contrôler le nombre de littéraux qui seront ajoutés à une clause. Par exemple, la spécification $P_{lit.}(\#littéraux = 2) = 1$ force toutes les clauses du langage à contenir exactement deux littéraux dans leur corps. De même, la spécification

$$P_{lit.}(\#littéraux = 1) = P_{lit.}(\#littéraux = 2) = P_{lit.}(\#littéraux = 3) = \frac{1}{3}$$

[12] La précision par rapport à un concept cible est calculée par le rapport du nombre d'exemples positifs de ce concepts couverts par la clause sur le nombre total d'exemples.

[13] On peut également envisager d'autres formes de contraintes, suivant la nature du domaine ($T \neq v$, $T \geq v$, etc).

entraînera la construction d'un langage de description dans lequel il devrait y avoir sensiblement autant de clauses constituées de 1 littéral dans le corps, que de clauses constituées de 2 ou de 3 littéraux.

Spécification des domaines et ajout de contraintes

L'ajout de contraintes permet de spécialiser une clause et de la rendre ainsi plus discriminante. Par exemple, si l'on considère la clause $p(X_1) \leftarrow q(X_2, Y, Z)$ il est probable que cette clause soit peu discriminante pour un ensemble quelconque d'exemples positifs et négatifs du concept cible. En revanche, la clause $p(X_1) \leftarrow q(X_1, Y, Z), Z = v$ est une clause spécifique pouvant avoir un intérêt pour distinguer des exemples.

Par défaut, nous choisissons d'associer systématiquement une contrainte, à chaque nouveau littéral ajouté. On distingue deux types de contraintes, possibles à ajouter : les contraintes de liaison (ou unifications) ou les contraintes génériques (numériques ou symboliques). Ces contraintes portent sur des variables et dépendent donc du type de domaine des variables. On décline alors trois types de domaines : les domaines de *liaison forcée*, les domaines de *liaison possible* et les domaines de *contrainte générique*.

Supposons, dans notre exemple, que :
- D_1 soit un domaine de liaison forcée (on le note alors D_1^{force}),
- D_2 un domaine de liaison possible ($D_2^{poss.}$),
- D_3, D_4 et D_5 des domaines de contrainte générique ($D_3^{gener.}, D_4^{gener.}, D_5^{gener.}$).

Nous rappelons ici les typages des prédicats, en utilisant cette nouvelle notation :

$$p(D_1^{force}) \;\; ; \;\; q(D_1^{force}, D_2^{poss.}, D_3^{gener.}) \;\; ; \;\; r(D_2^{poss.}, D_4^{gener.}, D_5^{gener.})$$

Domaine de liaison forcée : L'ajout d'un littéral dont le domaine de l'une des variables est du type "domaine de liaison forcée", entraîne une unification de la variable concernée avec une variable de même domaine déjà présente dans la clause. Par exemple, partant de la clause $p(X_1) \leftarrow$, si on choisit d'ajouter le littéral $q(X_2, Y, Z)$[14], on oblige l'unification des variables X_1 et X_2 qui sont toutes deux du même domaine de liaison forcée D_1^{force}. Dans un premier temps, la clause est alors de la forme

$$p(\mathbf{X_1}) \leftarrow q(\mathbf{X_1}, Y, Z)$$

Cette première contrainte est systématiquement complétée par une seconde contrainte, parmi les deux autres types de contraintes, associées aux deux autres types de domaines (liaison possible et contrainte générique).

Domaine de liaison possible : une contrainte d'unification peut être ajoutée, sur une variable d'un domaine de liaison possible. Cette contrainte consiste, lorsque c'est possible, à unifier cette variable avec une variable déjà présente dans la clause en construction. Par exemple, étant donnée la clause obtenue précédemment ($p(X_1) \leftarrow q(X_1, Y, Z)$), aucune contrainte de ce type ne peut être ajoutée. En effet, la seule variable possible à unifier est Y (de domaine $D_2^{poss.}$), mais aucune autre variable de son domaine n'est présente.

Par contre, si l'on considère la clause suivante en construction :

$$p(X_1) \leftarrow q(X_1, Y_1, Z), Z = v$$

[14] Le choix du prédicat est effectué aléatoirement.

et que l'on choisisse d'ajouter le littéral $r(Y_2, T, U)$, la variable Y_2 (de domaine $D_2^{poss.}$) peut être unifiée à la variable Y_1 de même domaine et déjà présente, pour former la clause

$$p(X_1) \leftarrow q(X_1, \mathbf{Y_1}, Z), r(\mathbf{Y_1}, T, U), Z = v$$

Domaine de contrainte générique : une contrainte peut porter sur une variable de domaine de contrainte générique. Ces contraintes correspondent aux c_j de la forme $Z = v,\ Z \neq v,\ Z \leq v$, etc. avec v appartenant au domaine de Z.

Pour résumer, lorsqu'un littéral est ajouté à une clause en construction, on regarde d'abord si une unification est possible sur deux variables d'un même domaine de liaison forcé. Si tel est le cas, l'unification est effectuée. Dans tous les cas, une contrainte d'unification ou une contrainte générique est ensuite ajoutée.

Pour choisir d'ajouter plutôt l'une ou l'autre des deux types de contraintes, ainsi que la variable sur laquelle elle doit porter, on se réfère aux types et aux poids des domaines. En effet, à chaque domaine D_i est associé une probabilité $P_{dom.}(D_i)$, pouvant être considéré comme un poids. Le choix du type de contrainte dépend du type de domaine sélectionné (contrainte d'unification pour un domaine $D^{poss.}$ et contrainte générique pour un domaine $D^{gener.}$) et la sélection du domaine est effectuée en fonction de la distribution $P_{dom.}$.

Considérons, par exemple, la distribution suivante sur les domaines de notre exemple[15] :

$$P_{dom.}(D_2^{poss.}) = 0.25 \ \ ; \ \ P_{dom.}(D_3^{gener.}) = 0.5$$

$$P_{dom.}(D_4^{gener.}) = 0.125 \ \ ; \ \ P_{dom.}(D_5^{gener.}) = 0.125$$

Supposons que $p(X_1) \leftarrow q(X_1, Y_1, Z), Z = v$ soit la clause en construction et $r(Y_2, T, U)$ le littéral à ajouter. Aucune des variables Y_2, T ou U n'appartient à un domaine de liaison forcée, il n'y a donc pas d'unification "forcée" à effectuer. Par contre, par défaut, une contrainte doit être ajoutée sur l'une des trois variables Y_2, T ou U, de domaines respectifs $D_2^{poss.}$, $D_4^{gener.}$ et $D_5^{gener.}$. Étant donnée la distribution de probabilité proposée ci-dessus, le domaine de Y_2 a deux fois plus de chance d'être choisi que les domaines de T ou de U, ainsi, l'une des trois clauses suivantes est construite, avec les probabilités associées :

$P(p(X_1) \leftarrow q(X_1, \mathbf{Y_1}, Z), r(\mathbf{Y_1}, T, U), Z = v) = 2/3$ (choix du domaine $D_2^{poss.}$),
$P(p(X_1) \leftarrow q(X_1, Y_1, Z), r(Y_2, \mathbf{T}, U), Z = v_1, \mathbf{T} = \mathbf{v_2}) = 1/3$ (choix du domaine D_3^{gener}),
$P(p(X_1) \leftarrow q(X_1, Y_1, Z), r(Y_2, T, \mathbf{U}), Z = v_1, \mathbf{U} = \mathbf{v_2}) = 1/3$ (choix du domaine D_4^{gener}).

Contraintes additionnelles

Un dernier paramètre influant pour la construction des clauses du langage de description est le nombre de contraintes additionnelles. En effet, une fois les littéraux ajoutés (en nombre spécifié), k contraintes sont ajoutées. Ces contraintes sont générés de la même façon que précédemment, en choisissant un domaine relativement à la distribution $P_{dom.}$ puis en ajoutant la contrainte liée au domaine (unification ou générique). Le nombre k est, quant à lui, défini par une nouvelle distribution $P_{contr.}$.

[15] Notons qu'il est inutile de poser un poids pour les domaines de liaison forcée, pour lesquels l'unification est systématique (e.g. $D_1^{poss.}$).

Par exemple, si on choisit la distribution

$$P_{contr.}(\#contraintes = 0) = P_{contr.}(\#contraintes = 1) = 0.5,$$

les clauses du langage contiendront au plus 1 contrainte additionnelle et il y aura approximativement autant de clauses qui ne contiennent pas de contrainte additionnelle que de clauses qui en contiennent une.

Nous proposons dans la section suivante, des expérimentations avec différents langages sur la base de données Mutagénèse. Nous observons à cette occasion l'influence des différents paramètres $|\mathcal{L}|$, $P_{dom.}$ et $P_{lit.}$ du langage, sur la qualité de la mesure induite.

4.5.2 Expérimentations

L'évaluation de la mesure de similarité est effectuée sur deux exemples traditionnels en ILP : Ancêtres et Mutagénèse.

Exemple 3 : Ancêtres

Cet exemple a été proposé initialement dans [41]. On cherche à apprendre le concept *ancêtre*, à partir d'une base de connaissances contenant les symboles de prédicat *père*, *mère*, *masculin* et *féminin*, sur une famille constituée de 19 personnes produisant 361 exemples dont 56 positifs. Une définition habituelle de ce concept correspond aux 4 clauses suivantes :

1 : *ancêtre(X, Y)* ← *mère(X, Y).*
2 : *ancêtre(X, Y)* ← *mère(X,Z)*, *ancêtre(Z, Y).*
3 : *ancêtre(X, Y)* ← *père(X, Y).*
4 : *ancêtre(X, Y)* ← *père(X,Z)*, *ancêtre(Z, Y).*

correspondant aux 4 sous-concepts usuels : *"mère"*, *"ancêtre féminin"*, *"père"* et *"ancêtre masculin"*.

Pour calculer la similarité entre deux exemples, on considère le langage contenant toutes les clauses ayant *ancêtre(X, Y)* comme tête et au plus deux littéraux dans le corps. Ce langage est fini et de taille raisonnable (532 clauses) : toutes les clauses sont donc générées. La matrice de similarité est présentée en figure 4.4 : chaque ligne (resp. colonne) correspond à un exemple positif. Pour observer les régularités dans cette matrice, les lignes (resp. colonnes) sont organisées de façon à grouper dans la matrice, les exemples associés à un même sous-concept.

Dans la matrice (figure 4.4), les valeurs supérieures à un seuil (que l'on fera varier) sont grisées ; ces valeurs correspondent aux couples d'exemples similaires. Si ce seuil est réduit, la proportion grisée augmente, comme le montre la figure 4.5 (le seuil dans la matrice (a) est supérieur au seuil de la matrice (b), lui même supérieur au seuil de la matrice (c)). Dans cette figure, on peut voir clairement que la similarité obtenue entre les exemples d'un même sous-concept de référence est élevée et que les exemples des deux sous-concepts "ancêtre féminin" et "ancêtre masculin" sont assez similaires. Ces observations correspondent à la décomposition obtenue avec PoBOC.

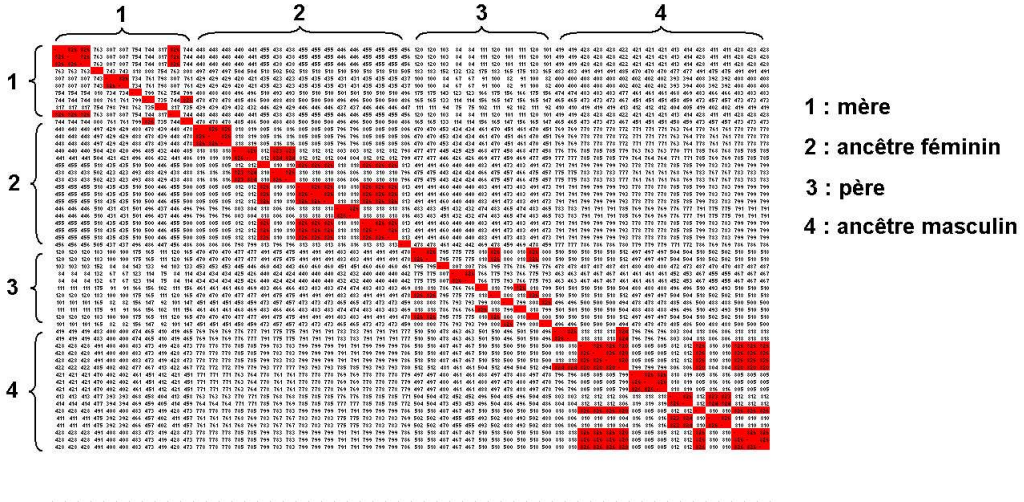


FIG. 4.4 – Matrice de similarité avec valeurs élevées grisées.

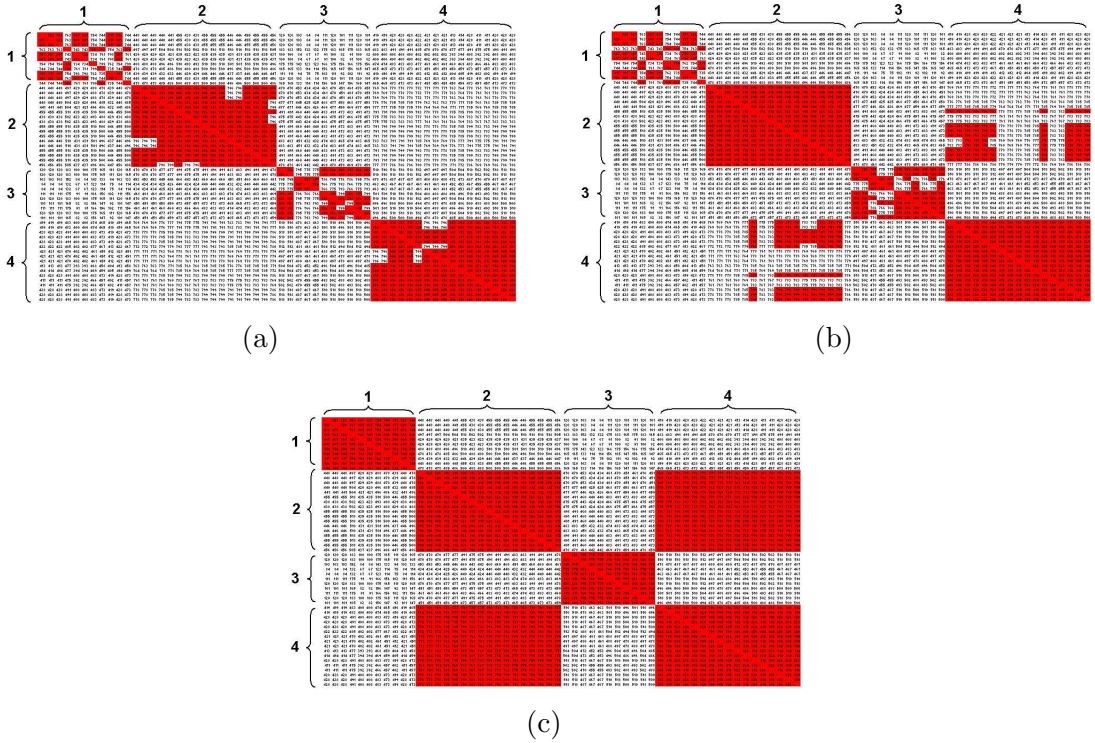


FIG. 4.5 – Matrices de similarité avec valeurs élevées grisées (variation du seuil).

Exemple 4 : Mutagénèse

Le problème associé à la base de données Mutagénèse[16] est la prédiction du caractère mutagène de composés, décrits en logique du premier ordre. Dans ce problème, on ne connaît pas de décomposition naturelle en sous-concepts. Nous choisissons alors d'évaluer la mesure de similarité en observant le taux d'exemples bien classés par une approche "1-Plus Proche Voisin" (1-PPV).

Les résultats que nous présentons sont obtenus par validations croisées. Pour une validation croisée, l'ensemble des exemples est scindé en 10 échantillons égaux (contenant chacun 10% de la base totale), chaque échantillon est choisi succéssivement comme ensemble de test. L'évaluation sur un échantillon consiste à affecter chaque exemple à la classe de son plus proche voisin, parmi les exemples d'entraînement (90% restant). De plus, cet exemple se place dans le cadre des langages infinis, présentés en section 4.5.1. Nous recherchons empiriquement les paramètres appropriés pour générer ce langage.

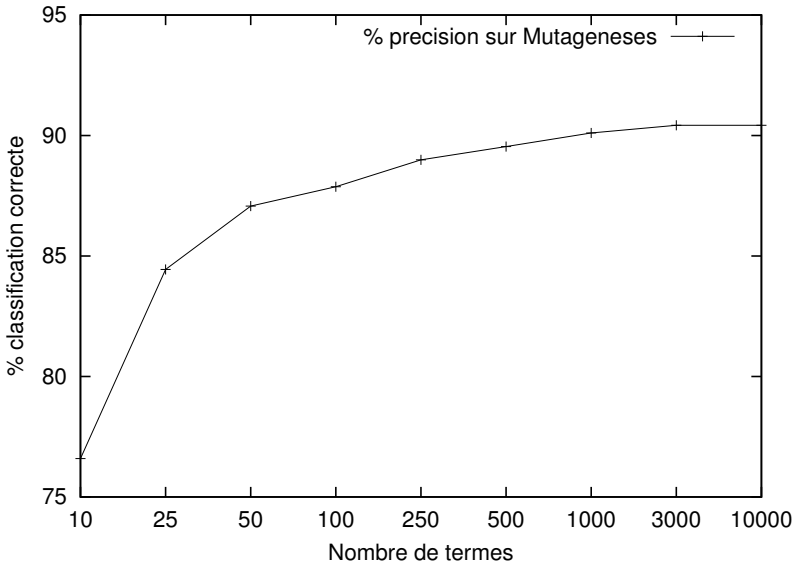


FIG. 4.6 – Performance du classifieur 1-PPV, pour des langages de tailles différentes.

Taille du langage : La première expérimentation permet d'observer l'influence de la taille du langage sur la précision du classifieur 1-PPV. Dans ce test, P_{dom} correspond à la probabilité uniforme, P_{lit} et P_{contr} sont définis par :

$$P_{lit}(\#littéraux = 1) = P_{lit}(\#littéraux = 2) = 0.5$$
$$P_{lit}(\#littéraux > 2) = 0$$
$$P_{contr}(\#contraintes = 0) = P_{contr}(\#contraintes = 1) = 0.5$$
$$P_{contr}(\#contraintes > 1) = 0$$

Les résultats suivants sont obtenus en effectuant une moyenne sur 10 exécutions de validations-croisées (figure 4.6). La précision augmente rapidement et les meilleurs

[16]La base de donnée *Mutagenesis* [176] est une base de référence dans le domaine de la programmation logique inductive.

résultats (90.43% de bonne classification) sont obtenus avec des langages contenant au moins 3000 clauses. Le résultat (90.11%) obtenu pour 1000 clauses est proche du résultat précédent. Dans un soucis de rapidité, nous choisirons par la suite de considérer des langages constitués de 1000 clauses.

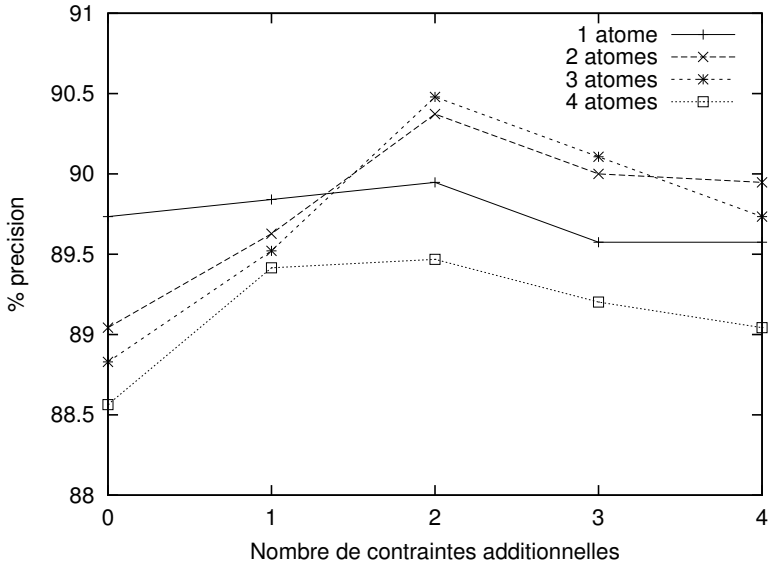


FIG. 4.7 – Performance du classifieur 1-PPV, pour différentes complexités de clauses.

Complexité des clauses. Ce test concerne la complexité (syntaxique) des clauses utilisées pour définir la similarité. Rappelons que cette complexité est principalement contrôlée par les distributions de probabilité P_{lit} et P_{contr} permettant de spécifier respectivement le nombre de littéraux et de contraintes additionnelles (ou unifications) dans la clause. Dans chacun des tests suivants, on génère des clauses à partir des schémas de distribution suivants : $P_{lit}(\#littéraux = i) = 1$, $P_{lit}(\#littéraux \neq i) = 0$, $P_{contr}(\#contraintes = j) = 1$ et $P_{contr}(\#contraintes \neq j) = 0$ pour $i = 1 \ldots 4$ et $j = 0 \ldots 4$.

Ces expérimentations (figure 4.7) montrent que quelque soit le nombre de littéraux dans le corps de la clause, la précision du classifieur est optimale quand 2 contraintes additionnelles sont ajoutées. Le meilleur score (90.48%) est obtenu avec 2 littéraux et 2 contraintes additionnelles.

Il est également intéressant de noter que lorsque les clauses du langage ne sont pas toutes de même complexité[17], les résultats obtenus sur la classification sont meilleurs que lorsque toutes les clauses du langage suivent un même schéma de construction (ce qui est le cas dans l'expérimentation actuelle). Dans les tests portant sur la taille du langage, les clauses construites pouvaient avoir : (1 littéral et 0 contrainte), (1 littéral et 1 contrainte), (2 littéraux et 0 contrainte) ou encore (2 littéraux et 1 contrainte). En considérant un seul schéma de clause, la précision est inférieure à 90% tandis qu'en autorisant un mélange des configurations précédentes, nous avons pu obtenir 90.43% de bonne classification.

[17]Dans l'expérimentation précédente portant sur la taille du langage, les complexités syntaxiques variaient d'une clause à une autre, pour un même langage.

Distribution de probabilités sur les domaines. Nous proposons à présent de tester l'influence des différents domaines apparaissant dans la base de données Mutagénèse. Nous modifions les poids de chaque domaine indépendamment dans P_{dom} et comparons la précision du classifieur 1-PPV, par rapport à la distribution par défaut (distribution uniforme). Pour chaque domaine, on évalue le classifieur avec un poids nul sur le domaine (ce qui revient à effectuer la classification sans ce domaine) puis un poids de 0.25 (les autres poids sont identiques et égaux à 0.075, le domaine considéré possède alors une plus grande influence sur la classification).

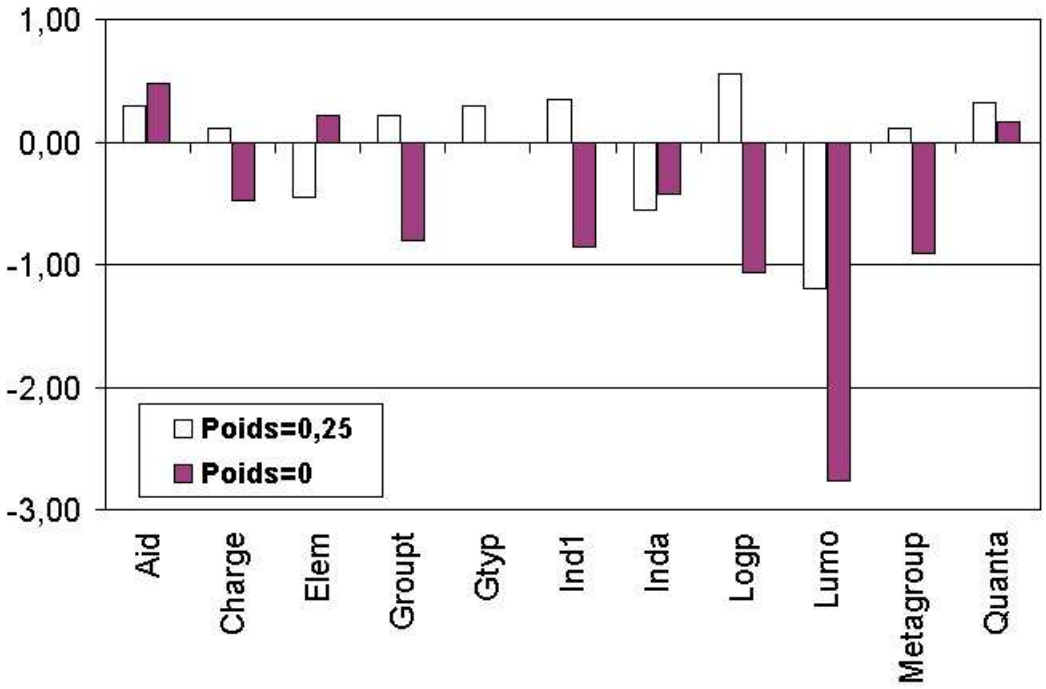


FIG. 4.8 – Performance du classifieur 1-PPV, pour différentes pondérations des domaines.

Cette expérience montre (figure 4.8) que certains domaines sont importants (comme par exemple *logp, Ind1, charge*) puisque la précision est plus faible lorsque le poids est nul et plus élevée lorsque le poids est de 0.25, par rapport à la distribution uniforme. En revanche, le domaine *elem* réduit la précision du classifieur. Enfin, pour d'autres domaines (*lumo, Inda*), la précision est plus faible dans les deux cas : on ne peut alors pas conclure sur leur importance indépendamment.

Pour conclure cette expérience, on teste le classifieur avec une distribution spécifique P_{dom} telle que les poids des domaines *logp, Ind1* et *charge* sont les plus élevés, le poids du domaine *elem* est le plus faible et les autres domaines ont des poids (intermédiaires) identiques. Ce paramétrage correspond à la configuration "optimale" dégagée empiriquement des quelques expérimentations précédentes.

On obtient alors, sur les 10 validations croisées, une moyenne de 91.915% de bonne classification. Ce résultat montre que la mesure de similarité que nous proposons est performante et que la précision du classifieur par plus proche voisin, pourrait sans doute

encore être améliorée en recherchant, de façon plus précise, les paramètres appropriés (distributions de probabilités).

4.6 Vue générale de la méthode

L'algorithme général d'apprentissage est présenté en figure 4.9. Les entrées de la méthode sont : le concept cible, spécifié par des exemples positifs et négatifs, une connaissance de base ainsi qu'un langage associé à la mesure de similarité. Nous supposons que le concept cible ne peut pas être caractérisé par une unique clause (dans le cas contraire, l'algorithme retourne simplement cette clause).

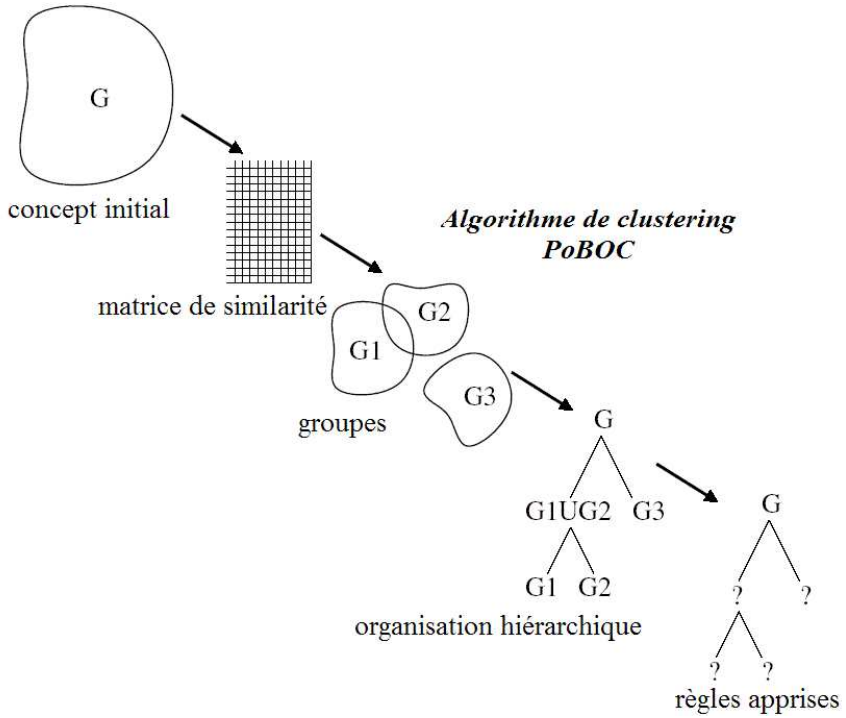


FIG. 4.9 – Vue générale de l'algorithme d'apprentissage disjonctif par décomposition.

La première étape de la méthode est le calcul de la similarité pour chaque paire d'exemples positifs. Ensuite, cette matrice de similarité est utilisée par l'algorithme de clustering PoBOC, afin de produire un ensemble de groupes, éventuellement non-disjoints. Dans certains cas, cet algorithme retourne un nombre important de groupes, certains d'entre eux étant assez similaires. Pour cette raison, nous organisons ces groupes hiérarchiquement : nous utilisons un algorithme agglomératif hiérarchique produisant un arbre (binaire) où les feuilles sont les groupes obtenus par la méthode de clustering et la racine de cet arbre représente le concept initial complet.

Ensuite, la méthode tente de construire une clause qui caractérise les groupes (nœuds) de l'arbre, en partant des deux sous-groupes de la racine : si une clause est trouvée pour un groupe, cette clause est ajoutée au programme appris ; s'il n'y a pas de clause apprise pour un groupe, ce processus est répété récursivement sur les sous-groupes directs du nœud correspondant (si ce nœud n'est pas une feuille). Finalement, si des groupes n'ont

pas été caractérisés par une clause, la méthode d'apprentissage (figure 4.9) est appliquée récursivement sur ces groupes.

Dans ce qui suit, nous discutons de la stratégie utilisée pour apprendre une clause et de la complexité globale du système. Enfin, nous proposons des expérimentations sur différents exemples.

4.6.1 Apprentissage d'une clause

Comme nous l'avons mentionné précédemment, la méthode de décomposition engendre une réduction forte de l'espace de recherche, puisque nous recherchons une seule règle, caractéristique d'un sous-concept. Étant donnés un sous-ensemble d'exemples positifs G (sous-concept) et l'ensemble des exemples négatifs E^-, on tente de construire une clause qui couvre tous les exemples de G et aucun exemple de E^-. La technique de construction que nous employons est une approche gloutonne. À partir de la clause la plus générale $p(X_1, \ldots, X_n) \leftarrow$ (où p est le prédicat cible, d'arité n), les littéraux sont ajoutés un par un jusqu'à ce que chaque exemple négatif soit rejeté par la clause. Puisque la clause doit couvrir tous les exemples de G, nous choisissons le littéral qui permet de rejeter un maximum d'exemples négatifs, parmi les littéraux qui couvrent tous les exemples de G.

4.6.2 Complexité de la méthode globale

On peut distinguer deux types de complexités : la complexité en temps et la complexité en espace. Cette dernière correspond à l'espace de stockage nécessaire à l'algorithme.

Complexité en temps

La complexité en temps, de la méthode, dépend des trois principales étapes de l'algorithme :

Similarité : pour calculer la matrice de similarité, nous devons d'abord tester si chaque exemple est couvert ou non par chaque clause du langage $\mathcal{L}$. Cette évaluation nécessite donc $(|E^+| + |E^-|) * |\mathcal{L}|$ tests de couverture ; la similarité est ensuite calculée pour chaque paire d'exemples positifs, ce qui nécessite $|E^+|^2 * |\mathcal{L}|$ opérations.

Clustering : l'utilisation de l'algorithme PoBOC détermine la complexité de cette étape, nécessitant alors au plus $k.|E^+|^2$ opérations, où k est le nombre de groupes obtenus.

Construction des clauses : la complexité de l'approche gloutonne est réduite par la restriction de l'espace de recherche.

Complexité en espace

La complexité en espace est déterminée par le nombre d'exemples positifs, puisque nous devons calculer une matrice de similarité de taille $|E^+| \times |E^+|$. Si $|E^+|$ est trop important (plus de 1000 exemples), on peut utiliser un échantillon de cet ensemble : le nombre d'exemples positifs nécessaires dépend principalement du nombre de clauses apprises. Par exemple, supposons que le concept cible puisse être caractérisé par 10 clauses, avec 1000 exemples positifs, nous avons encore une moyenne de 100 exemples par groupe.

4.6.3 Expérimentations

Nous proposons, dans cette section, quelques expérimentations de notre approche : nous testons la méthode d'apprentissage par décomposition avec PoBOC, sur des exemples pour lesquels une décomposition "naturelle" en sous-concepts est connue.

Exemple 1 : sous-concepts non-disjoints I (suite)

La première expérimentation concerne l'exemple 4.4.3. Si on utilise la matrice de similarité w_sim_α par rapport au langage $\mathcal{L}$ proposé dans l'exemple 4.4.3, nous obtenons des matrices de similarité proches de celles présentées dans l'exemple. Avec la première matrice nous obtenons les deux sous-concepts attendus $\{p(a), p(b)\}$ et $\{p(c), p(d)\}$; la seconde matrice donne également les 2 sous-concepts attendus $\{p(a), p(b), p(e)\}$ et $\{p(e), p(c), p(d)\}$ ($p(e)$ appartient aux deux sous-concepts).

Or, comme nous l'avons déjà montré, ces deux sous-concepts sont facilement caractérisés par un algorithme glouton. Les deux clauses obtenues sont celles attendues :

$C_1 : p(X) \leftarrow r(X), s(X),$
$C_2 : p(X) \leftarrow t(X), u(X).$

Exemple 2 : description numérique (suite)

La seconde expérimentation concerne l'exemple 4.4.4. Si on considère la similarité associée au langage

$$\mathcal{L}_1 = \bigcup_{v=1...6} (p(X) \leftarrow x(X,Y), \geq (Y,v)) \cup \bigcup_{v=1...6} (p(X) \leftarrow y(X,Y), \geq (Y,v)),$$

on obtient les sous-concepts attendus : $\{p(a), p(b), \ldots, p(d)\}$ et $\{p(e), p(f), \ldots, p(j)\}$. Si on considère à présent la similarité associée au langage $\mathcal{L}_2$ contenant toutes les clauses ayant $p(X)$ comme tête et au plus deux littéraux dans le corps, l'algorithme de clustering renvoie un résultat différent ; les groupes obtenus sont :

$G_1 = \{p(a), p(b), p(c), p(d)\}$ $\quad$ $G_2 = \{p(e), p(f), p(g), p(i)\}$ $\quad$ $G_3 = \{p(g), p(h), p(i), p(j)\}$

L'explication est basée sur la différence entre les deux langages $\mathcal{L}_1$ et $\mathcal{L}_2$: la clause $p(X) \leftarrow x(X,Y), y(X,Y)$ appartient à $\mathcal{L}_2$ mais non à $\mathcal{L}_1$. Cette clause est correcte puisqu'elle couvre 4 exemples positifs : $\{p(a), p(d), p(h), p(i)\}$ et aucun exemple négatif. Elle est donc pondérée par 1 dans $\mathcal{L}_2$. La conséquence est que la similarité entre $p(h)$ et $p(i)$ est augmentée et que, par exemple, la similarité entre $p(e)$ et $p(h)$ est réduite.

Cet exemple montre que lorsque différentes décompositions peuvent exister, cela peut induire une fragmentation de certains sous-concepts (particulièrement quand le nombre d'exemples est faible). Pour cette raison, le résultat de notre algorithme de pseudo-partitionnement est complété par une étape de hiérarchisation des clusters obtenus, avant d'apprendre une définition. Dans cet exemple, G_2 et G_3 sont les groupes les plus similaires, le résultat de la hiérarchisation est alors :

E^+ divisé en G_1 et $G_{2,3}$,
$G_{2,3}$ divisé en G_2 et G_3

où le premier niveau de décomposition correspond au résultat attendu.

Exemple 3 : Ancêtres (suite)

Sur l'ensemble complet, constitué de 56 exemples positifs, notre algorithme de clustering produit trois groupes disjoints :

G_1 : ce groupe correspond à l'ensemble $\{ancêtre(X_i, Y_i) | X_i$ est le père de $Y_i\}$
G_2 : ce groupe correspond à l'ensemble $\{ancêtre(X_i, Y_i) | X_i$ est la mère de $Y_i\}$
G_3 : ce groupe contient tous les autres exemples.

Les groupes les plus similaires sont G_2 et G_3, mais aucune clause caractérisant $G_2 \cup G_3$ ne peut être trouvée. L'algorithme tente alors de construire une caractérisation pour G_1, G_2 et G_3. Une clause est apprise pour G_1 et pour G_2, mais non pour G_3. Ensuite, l'algorithme de clustering est appliqué sur le groupe G_3, produisant deux groupes disjoints associés aux définitions récursives suivantes :

$ancêtre(X, Y) \leftarrow père(X, Z), ancêtre(Z, Y),$
$ancêtre(X, Y) \leftarrow mère(X, Z), ancêtre(Z, Y).$

Finalement, la décomposition aura produit 4 groupes, correspondant à la définition usuelle d'ancêtre.

Exemple 5 : sous-concepts non-disjoints II

Le dernier exemple est introduit pour tester la capacité de l'algorithme à construire des groupes non-disjoints. Considérons un graphe contenant deux types d'arêtes r et s. Dans la base de connaissances, l'atome clos $r(a, b)$ (resp. $s(a, b)$), exprime le fait qu'une arête de type r (resp. s), existe entre a et b. Cette base de connaissances contient l'ensemble suivant d'atomes clos, listés et représentés graphiquement dans la figure 4.10.

base de connaissance :

$\{r(b, f), r(b, g), r(f, l),$
$r(g, m), r(k, r), r(l, o),$
$r(m, p), r(m, o), r(j, n),$
$r(a, f), r(e, k), s(a, f),$
$s(e, k), s(j, n), s(c, g),$
$s(c, h), s(c, i), s(d, i),$
$s(d, j), s(h, m), s(h, n),$
$s(i, n), s(n, q), s(h, l)\}$

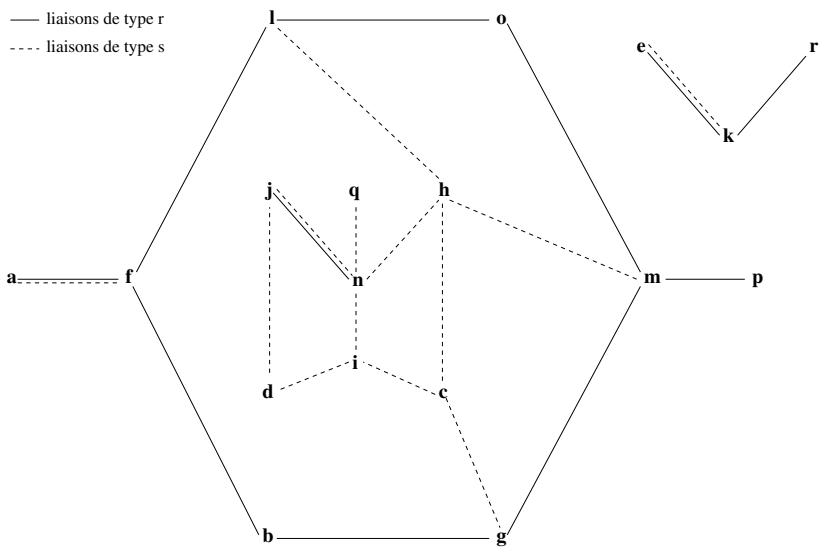


FIG. 4.10 – base de connaissance (à gauche) et représentation graphique (à droite).

Le concept cible est le concept *liés*, tel que *liés*(X, Y) est vrai si un chemin existe de X à Y (quelque soit la nature des arcs). Pour calculer la similarité entre deux exemples, on

considère le langage contenant toutes les clauses ayant *liés*(X, Y) comme tête, et au plus deux littéraux dans le corps. Dans cet exemple, il existe des sommets liés par différents chemins. A partir de l'ensemble des 44 exemples positifs, l'algorithme de clustering produit 5 groupes G_1, G_2, G_3, G_4 et G_5.

- G_1 et G_2 ont trois exemples en commun : G_1 est défini par *liés*$(X, Y) \leftarrow r(X, Y)$, et G_2 est défini par *liés*$(X, Y) \leftarrow s(X, Y)$. Il existe exactement trois exemples reliés directement par les deux types d'arêtes r et s (*liés*(a, f),*liés*(e, k) et *liés*(j, n)). G_1 et G_2 ne partagent pas d'autres objets avec les autres groupes.
- G_4 et G_5 ont un exemple en commun, et ces groupes sont les plus similaires. $G_4 \cup G_5$ est caractérisé par la clause *liés*$(X, Y) \leftarrow s(X, Z)$, *liés*$(Z, Y)$.
- G_3 ne partage aucun exemple avec d'autres groupes. Il est défini par la clause récursive *liés*$(X, Y) \leftarrow r(X, Z)$, *liés*$(Z, Y)$ et G_3 contient tous les exemples couverts par cette clause, à l'exception de certains exemples, qui sont également couverts par la clause récursive *liés*$(X, Y) \leftarrow s(X, Z)$, *liés*$(Z, Y)$.

Dans cet exemple, les intersections entre les groupes ne sont pas aussi importantes que l'on pourrait s'y attendre. Ceci est dû, principalement, aux spécificités qui augmentent la similarité entre certains exemples et réduisent la possibilité pour certains exemples d'être affectés (dans l'algorithme PoBOC) à plusieurs pôles, puisque cette fonction d'affectation est basée sur les similarités relatives. Cependant, il est préférable d'obtenir des groupes "incomplets" puisque pour chaque groupe, on tente d'apprendre une unique clause. Ainsi, la décomposition obtenue sur cet exemple est pertinente et permet de produire une définition satisfaisante du concept *liés*.

4.7 Conclusion

Dans ce chapitre, nous avons montré qu'il était possible d'étendre le système RuLe-Clust, à l'apprentissage de règles logiques. Cette extension nécessite cependant de spécifier un langage de description adapté pour définir une mesure de similarité pertinente sur ce type d'objets. Cette mesure a été évaluée et validée sur la base Mutagénèse, faisant référence dans le domaine de la programmation logique inductive.

La phase de décomposition utilise l'algorithme PoBOC, qui a pu être évalué sur quatre exemples pour lesquels une organisation naturelle des exemples positifs du concept cible, était connue. Sur chacun des exemples, PoBOC s'est révélé être performant, tant du point de vue du nombre de groupes retournés, que de la constitution même de ces groupes et des intersections proposées.

Sur la base Mutagénèse, PoBOC n'a pu être testé, en l'absence d'une stratégie efficace de construction de règles. En effet, la réduction de l'espace de recherche, engendrée par notre approche de décomposition, n'est pas suffisante pour permettre le test d'existence d'une clause caractéristique en temps raisonnable. Contrairement au cas propositionnel, on ne peut pas tester à l'avance l'existence d'une clause caractéristique pour un sous-concept donné, et la recherche d'une telle clause est coûteuse. Nous envisageons alors une stratégie basée sur le test de couverture des exemples négatifs par un plus petit généralisé d'exemples d'un groupe.

5

Classification de données textuelles pour la Recherche d'Information

Sommaire

5.1 Introduction à la Recherche d'Information 128
5.2 Mesure de similarité et données textuelles 129
5.2.1 Évaluer la similarité entre les mots 129
5.2.2 Les mesures basées sur les cooccurrences 129
5.2.3 Les mesure basées sur des distributions 130
5.2.4 Les mesures utilisant les connaissances extérieures 130
5.2.5 Un nouvel indice de proximité basé sur le Web 131
5.2.6 Expérimentations sur l'indice de proximité contextuelle 134
5.3 Classification de mots 137
5.3.1 Travaux antérieurs 137
5.3.2 PoBOC : un algorithme approprié ? 138
5.3.3 Expérimentation 139
5.4 Regroupement de mots pour la classification de documents . . 143
5.4.1 Introduction à la classification de documents 143
5.4.2 Classification supervisée de documents 143
5.4.3 Réduction de la dimension de l'espace de description des documents 146
5.4.4 Le classifieur naïf de Bayes pour la classification de documents . 149
5.4.5 Regroupement distributionnel agglomératif 150
5.4.6 Partitionnement par optimisation d'une fonction de qualité, issue de la théorie de l'information 152
5.4.7 Réduction de l'espace par pseudo-partitionnement du vocabulaire 152
5.4.8 Étude expérimentale 158
5.5 Étude prospective : application à l'analyse des genres 161

5.1 Introduction à la Recherche d'Information

Le domaine de la recherche d'information (*Information Retrieval*) englobe toutes les opérations à mettre en œuvre pour satisfaire des demandes d'information d'un utilisateur. Pour revenir sur une distinction classique dans le domaine, on oppose la recherche d'information à la recherche de documents sur des critères qui ont trait à la nature de l'information à traiter. L'information peut en effet être primaire ou secondaire.

L'information primaire est l'information d'origine issue des documents initiaux, qu'ils soient papier ou électronique, qu'il s'agisse de données textuelles ou non-textuelles (photographies, graphiques, schémas, vidéo, son, multimédia, etc.). Actuellement on a de plus en plus accès à des textes intégraux, mis à disposition par des bases de textes réglementaires, des bases de textes scientifiques, des sources de documentation technique de produits, des journaux (agence de presse), des pages Web, des informations portant sur la mémoire d'entreprise, etc.

L'information secondaire, quant à elle, s'applique aux ouvrages de référence qui sont des documents élaborés à partir des documents primaires. Ils ne contiennent pas de connaissances nouvelles. Relèvent de cette catégorie, les bibliographies, les bulletins d'analyse, les catalogues, les répertoires, etc. qui fournissent une description signalétique des documents primaires assortie d'une description de leur contenu.

De cette typologie résulte la distinction entre recherche documentaire et recherche d'information [109]. La recherche documentaire s'effectue dans des bases de données factuelles (bases de données bibliographiques par exemple) et s'applique à des données structurées numériques ou alphanumériques. La recherche d'information s'applique à des données primaires consignées dans des bases de données en texte intégral ou accessibles sur le Web. Les informations peuvent être structurées ou non.

On s'intéresse dans ce travail à la recherche d'information qui porte sur des gisements d'information contenant majoritairement du texte.

Deux opérations parallèles et complémentaires sont constitutives du processus de recherche d'information : l'*indexation* et l'*interrogation*.

L'*indexation* consiste à représenter le contenu des documents à partir de deux sources d'information : le document lui-même et le fonds auquel il appartient. Le résultat de l'indexation est une représentation des connaissances contenues dans les documents qui le constituent [103]. Cette représentation peut être plus ou moins élaborée. Les deux pôles extrêmes sont, au niveau le plus bas, une représentation à base de chaînes de caractères issues des documents. C'est par exemple, le cas des index constitués de manière automatique par les moteurs de recherche (à l'exclusion des mots vides entre autres). Au niveau le plus élevé, une représentation des connaissances plus élaborée permet par exemple, de produire des inférences sur des index conceptuels [146].

De l'autre côté, l'*interrogation* doit s'exprimer à travers la représentation des documents. L'expression des demandes est liée au degré d'interactivité des systèmes ; elle est prise en charge par une interface.

L'étude que nous abordons dans ce chapitre, porte sur la phase d'indexation. Nous nous focalisons dans un premier temps sur une représentation à partir de chaînes de caractères contenues dans les documents. Plus précisément nous nous intéressons successivement aux mesures permettant d'évaluer la proximité contextuelle de deux mots, puis aux méthodes de regroupement aboutissant à la formation de classes sémantiques, enfin nous utilisons

ces derniers éléments pour indexer les documents dans un processus de classification supervisée.

Pour terminer, nous tentons d'apporter quelques justifications aux travaux récents, qui consistent à représenter les informations textuelles à un niveau supérieur, en utilisant l'information morpho-syntaxique contenue dans les documents.

5.2 Mesure de similarité et données textuelles

5.2.1 Évaluer la similarité entre les mots

La quantification des relations entre termes est le sujet d'intenses recherches depuis plusieurs décennies, les applications sont très nombreuses. Le problème de la désambiguïsation sémantique peut être en partie traité en recherchant les mots susceptibles d'être trouvés à proximité d'un mot donné [65]. D'autres tâches de Traitement Automatique du Langage Naturel (TALN), comme par exemple l'acquisition (semi-)automatique d'un lexique sémantique [155] ou d'un thésaurus [72], utilisent la notion de proximité entre termes.

Nous nous intéressons, dans ce chapitre, aux relations contextuelles, que nous appellerons, parfois abusivement "relations sémantiques", dans une optique d'application à la Recherche d'Information. Connaître et quantifier les relations qui existent entre des termes permet par exemple, dans ce contexte de la RI, l'expansion des requêtes [147], l'indexation des documents [191] ou l'aide à la visualisation de l'information [37].

Plusieurs types d'approches ont été explorées, dans le but de définir des mesures de proximité contextuelle entre des termes [129]. Nous choisissons de les classer relativement aux informations qu'elles utilisent.

5.2.2 Les mesures basées sur les cooccurrences

Les mesures statistiques de cooccurrences sont très souvent utilisées pour découvrir les récurrences contextuelles. Ces mesures supposent l'existence d'une masse importante de données, par exemple, l'utilisation des mots ciblés dans un corpus de documents. Ces mesures se fondent sur l'hypothèse harrissienne selon laquelle "Les mots apparaissant dans des contextes similaires tendent à avoir des sens similaires" [78].

La mesure d'*Information Mutuelle* (IM) est l'une des premières mesures utilisées pour détecter ces récurrences [58]. Cette mesure compare la probabilité de trouver, dans un corpus, deux mots simultanément avec la probabilité de les trouver indépendamment :

$$IM(w_t, w_s) = \log_2 \frac{p(w_t, w_s)}{p(w_t)p(w_s)}$$

Cette mesure n'est pas définie pour $p(w_t, w_s) = 0$, et n'est pas un indice de similarité (la propriété de minimalité n'étant pas vérifiée). Plusieurs variantes de la mesure d'information mutuelle ont été proposées, en utilisant des fenêtres contextuelles[1] et en tenant compte de l'ordre d'apparition des mots (*Rapport d'association*) [23].

Remarquons que l'information mutuelle est relative à la taille du corpus utilisé pour mesurer les cooccurrences. Pour éviter ce phénomène [133] proposent la méthode des *mots associés* :

$$I_{SDOC}(w_t, w_s) = \frac{p(w_t, w_s)^2}{p(w_t)p(w_s)}$$

[1] La notion de "fenêtre contextuelle" permet de rechercher les mots qui apparaissent ensemble dans les mêmes phrases, ou séparés par un nombre limité de mots.

Il s'agit, cette fois, d'un indice de similarité (les propriétés de symétrie et minimalité étant vérifiées.

Enfin, le *coefficient de Dice* [43] introduit par [170] et la mesure de *Jaccard* [90], sont adaptés au contexte de l'analyse des proximités entre mots :

$$Dice(w_t, w_s) = \frac{2.p(w_t, w_s)}{p(w_t) + p(w_s)}$$

$$J(w_t, w_s) = \frac{p(w_t, w_s)}{p(w_t) + p(w_s) - p(w_t, w_s)}$$

La mesure de *Jaccard* (aussi appelée mesure de *Tanimoto*) correspond en fait au rapport du nombre de documents où w_t et w_s cooccurrent sur le nombre de documents où l'un des deux mots, au moins, apparaît. Ces deux dernières mesures sont également des indices de similarité.

Nous proposerons par la suite de comparer et d'adapter les mesures précédentes. Cette étude expérimentale débouchera sur la définition d'une nouvelle mesure, validée sur un exemple et permettant ainsi l'évaluation de l'algorithme PoBOC, sur cet exemple.

5.2.3 Les mesure basées sur des distributions

Les approches à base de cooccurrences permettent d'identifier les mots susceptibles d'être trouvés à proximité d'un mot donné. Ces mesures sont traditionnellement utilisées pour l'expansion de requêtes. Par exemple, si un utilisateur interroge une base documentaire en proposant une requête constituée de peu de mots, la recherche des documents contenant uniquement ce ou ces mots est propice au bruit[2]. L'expansion de la requête est un processus permettant de proposer (généralement sous le contrôle de l'utilisateur) un ensemble complémentaire de mots, proches du/des mot(s) de la requête initiale, afin de préciser cette dernière.

Cependant, ces mesures ne permettent pas de mettre en relation certains types de mots, tels que les synonymes, par exemple. En effet, deux mots synonymes sont rarement utilisés ensembles, il est encore plus rare de les observer à proximité les uns des autres (fenêtres contextuelles). Il s'agit, en fait, de mots qui n'apparaissent pas ensemble, mais dont les environnements sont identiques. L'approche distributionnelle permet alors de rechercher ces mots, apparaissant dans des contextes similaires.

Il s'agit alors de décrire les contextes, généralement syntaxiques, d'utilisation des mots. Par exemple, [158] analysent les relations à l'intérieur des syntagmes nominaux, eux même repérés par une analyse syntaxique. Plus précisément, les travaux de Agrawal [2] décrivent chaque mot en fonction des relations *sujet/verbe*, *verbe/complément* et *Préposition/nom*. Ainsi, un nom peut être décrit par 8 descripteurs : les 3 verbes dont il est le plus fréquemment sujet, les 3 verbes dont il est le plus fréquemment objet ainsi que les deux prépositions avec lesquelles il est le plus fréquemment associé. Les classes contextuelles sont ensuite obtenues par comparaison des vecteurs descriptifs.

5.2.4 Les mesures utilisant les connaissances extérieures

Une troisième approche pour évaluer la proximité entre mots, consiste à intégrer des connaissances, sinon syntaxiques, cette fois sémantiques, pour définir une mesure. Le plus

[2]Informations retournées faiblement pertinentes, relativement à la requête donnée.

souvent, ces connaissances sont formalisées par une taxonomie de concepts[3], dans laquelle sont analysées les positions respectives des termes à comparer, notamment par le repérage du concept le plus spécifique (C) contenant les deux termes.

Par exemple, [185] utilise les liens "IS-A" dans un telle taxonomie, en observant la "position"[4] de C, par rapport à la racine et aux termes cibles. Plus C est situé proche des termes cibles et éloigné de la racine, plus les termes seront considérés similaires.

[113] ou encore [154] se situent à un niveau plus général, en pondérant chaque concept de la taxonomie par la probabilité qu'un objet sélectionné aléatoirement appartienne à ce concept. La mesure de similarité obtenue tient compte notamment du poids associé au concept le plus spécifique (C) contenant les deux termes cibles.

La principale limite de ce type d'approche est que, le plus souvent, il n'existe pas de base de connaissance spécifique à un domaine et suffisamment structurée.

5.2.5 Un nouvel indice de proximité basé sur le Web

Nous proposons ici un nouvel indice, à partir duquel nous évaluerons la construction de classes contextuelles de mots, par l'algorithme PoBOC. Cet indice est inspiré des trois aspects présentés jusqu'ici, complétés par les travaux de P. Turney [180], utilisant le Web comme ressource terminologique.

Le Web peut être considéré comme la plus importante ressource en matière d'information textuelle. Bien qu'il soit techniquement difficile de caractériser ce "corpus" d'un point de vue sociolinguistique, il est cependant légitime de collecter les productions écrites par la "communauté Web", à partir du moment où elles éclairent la langue et la connaissance. Le Web est déjà utilisé pour de nombreuses applications en TAL, telles que l'enrichissement automatique d'ontologies [3] ou la génération de dictionnaires spécialisés [73]. P. Turney [180] est le premier à proposer d'évaluer la proximité sémantique entre les termes à partir de leur utilisation sur le Web. Les résultats obtenus pour la tâche d'identification de synonymes sont surprenants, et nous incitent à poursuivre dans cette voie.

Hormis l'avantage lié à la masse d'information disponible sur le Web, cette ressource présente la particularité de disposer, à l'avance, d'outils d'interrogation spécifiques, rapides et précis. En effet, il existe des moteurs de recherche thématiques, permettant de limiter l'analyse à un sous-ensemble de pages spécifiques et certains moteurs proposent des outils de recherche avancés afin de paramétrer, par exemple, l'analyse des cooccurences (opérateurs AND, NEAR, etc.).

Étant donné un ensemble de mots cibles (dont on cherche à apprendre les relations sémantiques/contextuelles), l'indice que nous proposons commence par analyser les cooccurrences des mots sur le Web afin d'évaluer les récurrences contextuelles, puis compare pour deux mots donnés, les deux vecteurs de récurrences contextuelles obtenus précédemment. Considérons, par exemple, l'ensemble de mots cibles suivant :

$$\{\textit{Salade, Restaurant, Asperge, Cour Suprême, Magistrat, Citoyen, Avocat}\}$$

Pour chacun de ces mots, on commence par rechercher ses cooccurrences avec les autres mots. En utilisant, par exemple, le moteur de recherche *Altavista*[5], on obtient la matrice

[3] Termes ou groupes de termes organisés hiérarchiquement afin de révéler les relations de généralité entre ces termes.

[4] Dans la mesure de [185], la position d'un concept A par rapport à un concept B est fonction du nombre de liens "IS-A" entre A et B dans la taxonomie.

[5] *www.altavista.com*

présentée en table 5.1.

hits	Salade	Restaurant	Asperge	Cour Suprême	Magistrat	Citoyen	Avocat
Salade	567000	-	-	-	-	-	-
Restaurant	114000	49900000	-	-	-	-	-
Asperge	11200	2240	83800	-	-	-	-
Cour Suprême	190	1410	97	173000	-	-	-
Magistrat	364	15600	198	2950	620000	-	-
Citoyen	1560	30800	534	22400	26800	948000	-
Avocat	26200	49500	2870	37700	53600	64200	114000

TAB. 5.1 – Nombre de pages Web retournées pour chaque requête de un ou deux mots.

On observe, dans cette matrice, une cohérence globale des valeurs. Les plus faibles scores sont obtenus pour les couples de mots (*Court Suprême, Asperge*), (*Court Suprême, Salade*) et (*Magistrat, Asperge*) qui correspondent effectivement à des mots que l'on aurait peu tendance à utiliser ensembles. En revanche, les plus fortes valeurs observées sont pour les couples (*Salade, Restaurant*), (*Avocat, Citoyen*), et (*Avocat, Magistrat*), que l'on imagine très bien apparaître dans les mêmes documents.

hits	Salade	Restaurant	Asperge	Cour Suprême	Magistrat	Citoyen	Avocat
Salade	-19,11	-27,89	-22,02	-28,94	-29,85	-28,36	-24,56
...							
Magistrat	-29,85	-30,89	-27,97	-25,12	-19,24	-24,39	-23,65
...							
Avocat	-24,56	-30,10	-24,99	-22,32	-23,65	-24,00	-20,12

TAB. 5.2 – Information Mutuelle pour chaque couple de mots.

L'évaluation des récurrences contextuelles peut être réalisée par l'une des mesures présentées dans la section 5.2.2. Nous proposerons, par la suite, une étude expérimentale pour choisir la mesure la plus appropriée. Par exemple, la mesure d'Information Mutuelle transforme la matrice de cooccurrences précédente, en matrice de proximités, présentée dans le tableau 5.2.

Dans ce tableau, les proximités révélées sont une nouvelle fois cohérentes ; si l'on construit pour chacun des trois mots *Salade, Magistrat* et *Avocat*, la liste ordonnée des mots avec lesquels ils sont le plus susceptibles d'apparaître, on obtient les trois listes suivantes :

Salade → *Salade, Asperge, Avocat, Restaurant, Citoyen, Cour Suprême, Magistrat*
Magistrat → *Magistrat, Avocat, Citoyen, Cour Suprême, Asperge, Salade, Restaurant*
Avocat → *Avocat, Cour Suprême, Magistrat, Citoyen, Salade, Asperge, Restaurant*

Ces listes doivent être interprétées de la façon suivante : "les mots *Asperge* et *salade* sont davantage susceptibles d'être observés dans un même document, que les mots *Salade* et *Magistrat*". Comme nous l'avons déjà mentionné, ce type de mesure, basée sur

les cooccurrences, ne permet pas de détecter les relations telles que la synonymie. Nous choisissons, pour aboutir à un indice de proximité contextuelle, d'analyser les distributions des mots sur l'ensemble de mots lui-même.

Pour ce faire, on considère chaque ligne de la matrice de proximité (tableau 5.2), comme un vecteur contextuel, associé à un mot. Par exemple, le vecteur contextuel du mot *Magistrat* est :

$$contexte(magistrat) = (-29.85, -30.89, -27.97, -25.12, -19.24, -24.39, -23.65)$$

Dans la tâche de regroupement contextuel de mots, que nous visons ici, la taille du vecteur correspond au nombre de mots à traiter. Chaque mot est décrit par sa proximité (susceptibilité d'apparaître dans les mêmes documents) avec les autres mots, et la proximité contextuelle entre deux mots peut être évaluée par comparaison des deux vecteurs caractéristiques. Chaque mot est en quelque sorte utilisé pour décrire les autres mots. Nous utilisons le coefficient de corrélation pour comparer deux vecteurs. Sur notre exemple, nous obtenons les coefficients suivants :

$I_{context}(Salade, Magistrat) = -0.56,$
$I_{context}(Salade, Avocat) = 0.05,$
$I_{context}(Avocat, Magistrat) = 0.64.$

Ces coefficients reflètent effectivement nos intuitions sémantiques sur les relations entre les trois mots comparés. Les mots *Salade* et *Magistrat* font référence à des concepts très différents et l'indice de proximité contextuel renvoie une valeur négative. En revanche, le mot *Avocat* est ambigu puisqu'il peut aussi bien signifier une "personne dont le métier est de défendre quelqu'un ou quelque chose" que le "fruit de l'avocatier" ; le mot *Avocat* est un homonyme qui appartient à la fois au concept de la *Justice* et à celui de la *Gastronomie*, ce qui ce manifeste par des indices de proximités contextuelles positifs.

Présentation formelle de l'indice de proximité contextuelle

Étant donné un ensemble $V = \{w_1, \ldots, w_l\}$ de mots, que l'on souhaite organiser en classes contextuelles/sémantiques. On note *hits(Q)*, le nombre de pages Web retournées par un moteur de recherche, pour une requête Q, constituée de un ou plusieurs mots de V.

Soit $I_{cooc.}$ un indice de proximité basé sur la notion de cooccurrences des mots sur le Web (*Information Mutuelle, indice des mots associés, coefficient de Dice, mesure de Jaccard,* etc.), on appellera *vecteur contextuel* d'un mot w_i, et on notera $\overrightarrow{w_i}$, le vecteur de $\mathbb{R}^l$ suivant :

$$\overrightarrow{w_i} = (I_{cooc.}(w_i, w_1), \ldots, I_{cooc.}(w_i, w_l))$$

avec $I_{cooc.}(w_i, w_j)$ calculé à partir des valeurs $hits(w_i)$, $hits(w_j)$ et $hits(w_i, w_j)$.

L'indice de proximité contextuelle que nous proposons ici, noté $I_{cont.}$, compare deux mots w_i et w_j en calculant le coefficient de corrélation sur les deux vecteurs correspondant $\overrightarrow{w_i}$ et $\overrightarrow{w_j}$:

$$I_{cont.}(w_i, w_j) = \rho(\overrightarrow{w_i}, \overrightarrow{w_j}) \quad \text{où} \quad \rho(\overrightarrow{x}, \overrightarrow{y}) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 . \sum_i (y_i - \bar{y})^2}}$$

Notons que cet indice ($I_{cont.}$) dépend du premier indice $I_{cooc.}$, dont les résultats dépendent, à leur tour, du moteur de recherche utilisé sur le Web. Dans les expérimentations

qui suivent, nous tenterons de déterminer empiriquement l'indice $I_{cooc.}$ le plus approprié. Concernant le choix du moteur, quelques tests préliminaires ont permis de dégager des résultats davantage pertinents avec les moteurs de recherche *Altavista* et *yahoo*[6], qu'avec *google*[7] ou encore *alltheweb*[8]. Cependant, nous ne présentons pas d'étude comparative sur ce sujet, qui pourrait faire l'objet d'un travail plus approfondi, notamment par l'analyse des conséquences des différentes méthodes d'indexation et d'interrogation par les moteurs, sur les proximités contextuelles.

5.2.6 Expérimentations sur l'indice de proximité contextuelle

Construction d'un jeu de données test

L'évaluation de l'indice de proximité contextuelle que nous venons de définir, nécessiterait le recours à un expert. Cependant, cette tâche reste difficile, même pour un expert, car très imprécise. En effet, il ne s'agit pas de confirmer ou non l'existence d'une relation (éventuellement typée), mais de donner un avis sur un ensemble de valeurs, qu'il est difficile d'appréhender dans son ensemble.

Nous allons contourner ce problème en collectant un ensemble de mots, dont les contextes d'utilisation peuvent servir de base de référence et être assimilés à l'expert. Notre expérience repose sur deux hypothèses, jugées raisonnables : on considère que les membres du comité de programme d'une grande conférence ou d'un journal sont parmi les meilleurs experts du domaine considéré ; nous supposons enfin que les mots-clés choisis par l'auteur d'un article de recherche sont représentatifs du contenu de ce papier. A partir de ces deux hypothèses, nous collectons un ensemble de mots-clés d'articles de recherche, et considérons que les mots issus d'un même domaine (représenté par une conférence ou un journal) sont plus similaires, au sens de l'indice de proximité contextuelle, que les mots provenant de domaines de recherche différents.

Un ensemble de 38 mots-clés provenant d'articles scientifiques publiés dans trois conférences ou journaux internationaux sont choisis arbitrairement. Les trois domaines sont :

Ressources du Langage : il s'agit de 11 mots-clés provenant d'articles publiés à l'occasion de la conférence LREC'2000 (*2nd International Conference on Language Resources and Evaluation*). La liste de ces mots est donnée dans le tableau 5.3.

Annotation Guidelines, Bracketed Corpus, Chinese Language Processing, Quality Control, Combining Systems, Machine Learning, Tagging, Knowledge-Rich NLP, Multilingual Corpora, Parallel Corpora, POS Tagging.

TAB. 5.3 – Les 11 mots-clés du domaine *Ressources du Langage*.

Web Mondial : ce domaine est représenté par 13 mots-clés, rencontrés dans des publications de la 11ème conférence internationale sur le Web Mondial (*11th International World Wide Web Conference*, WWW'2002). Ces mots sont répertoriés dans le tableau 5.4.

[6] *www.yahoo.com*
[7] *www.google.com*
[8] *www.alltheweb.com*

Content Distribution Networks, Data Consistency, Data Dissemination, Dynamic Data, HTTP, Leases, Protocol Design, Pull, Push, Scalability, TCP Splice, Web Proxy, World Wide Web.

TAB. 5.4 – Les 13 mots-clés du domaine *Web Mondial*.

Intelligence Artificielle : le dernier ensemble de mots-clés est issu des articles publiés dans la revue d'Intelligence Artificielle *Journal of Japanese Society for Artificial Intelligence* (JSAI'1997). le tableau 5.5 présente les 14 mots-clés retenus pour ce domaine.

Classification Rule, Macro Rule, Concept Learning, Constructive Induction, Colored Digraph, Logic Programming, Problem Solving, Program Transformation, Unfolding, Control of Computation, Natural Language Processing, Ill-Formedness, Robust Parsing, Integration.

TAB. 5.5 – Les 14 mots-clés du domaine *Intelligence Artificielle*.

Notons qu'il n'y a pas de restriction concernant la forme et la généralité des mots-clés : certains mots sont des unités polylexicales (*Content Distribution Networks*), d'autres sont monolexicales (*Tagging*) ; ils peuvent être assez généraux (*Push, integration*, etc.) ou au contraire spécifiques (*HTTP*).

Évaluation de l'indice

A partir de cet ensemble de mots-clés (noté V), l'évaluation d'une mesure de proximité entre mots consiste à comparer les valeurs de proximité pour les paires de mots d'un même domaine, avec les valeurs obtenues pour des paires de mots de domaines différents.

Les mesures comparées sont les suivantes :
- les indices $I_{cooc.}$, basés sur les cooccurrences uniquement : la mesure d'information mutuelle (IM), la méthode des mots associés (I_{SDOC}), le coefficient de Dice (I_{Dice}), la mesure de Jaccard (J),
- les indices $I_{cont.}$ de proximité contextuelle associés aux quatre indices précédents.

Toutes ces mesures sont évaluées en utilisant le moteur de recherche *Altavista*.

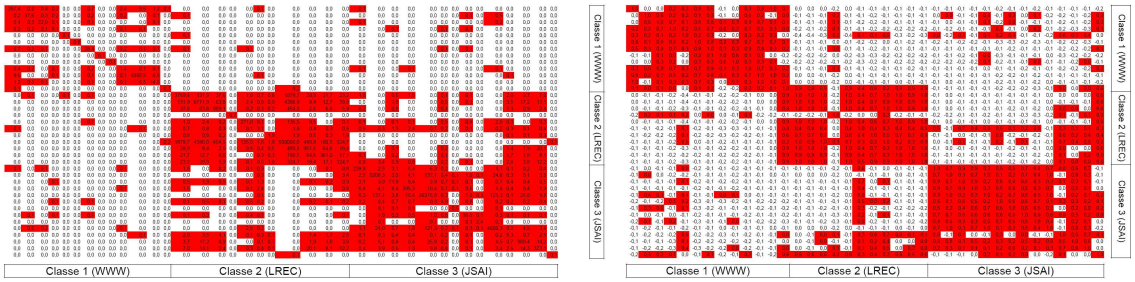
Les matrices de proximité correspondant à chacun de ces huit indices, sont présentées dans la figure 5.1. Les matrices indicées (a) correspondent aux indices $I_{cooc.}$ et les matrices indicées (b) à l'indice $I_{cont.}$ associé. Les couples (1,2,3, et 4) correspondent respectivement aux indices IM, I_{SDOC}, I_{Dice} et J. Par exemple, la matrice (3.a) contient les valeurs $\{I_{Dice}(w_i, w_j)\}_{w_i,w_j \in V}$ obtenues par le calcul suivant :

$$I_{Dice}(w_i, w_j) = \frac{2.hits(w_i, w_j)}{hits(w_i) + hits(w_j)}$$

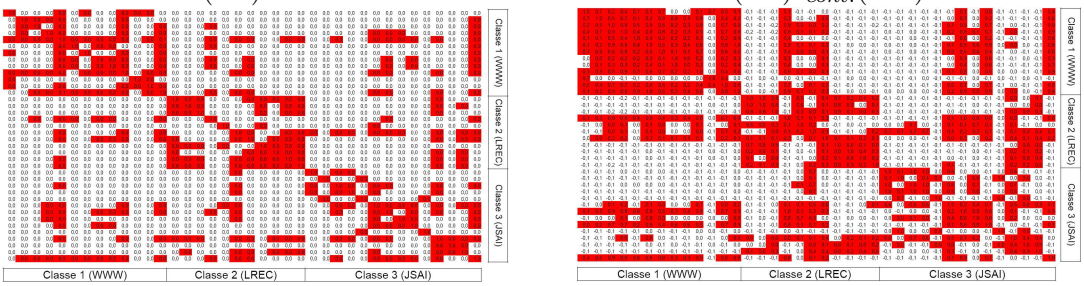
La matrice (3.b) contient les valeurs de l'indice de proximité contextuelle associé à I_{Dice}, $\{I_{cont.}(\overrightarrow{w_i}, \overrightarrow{w_j})\}_{w_i,w_j \in V}$ (cf. section 5.2.5).

Dans les matrices, les mots-clés sont organisés par domaine, constituant des blocs sur la diagonale. Les cases grisées correspondent aux valeurs de proximité élevées[9] dans le cas

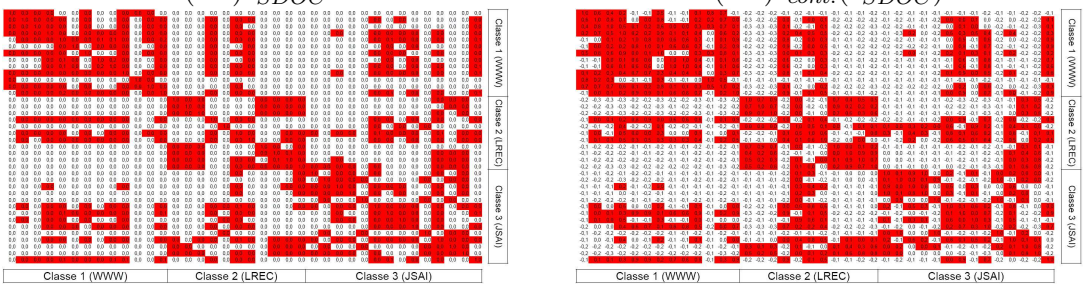
[9]Le seuil est choisi de telle manière que environ un tiers des cellules sont grisées (plus fortes proximités).



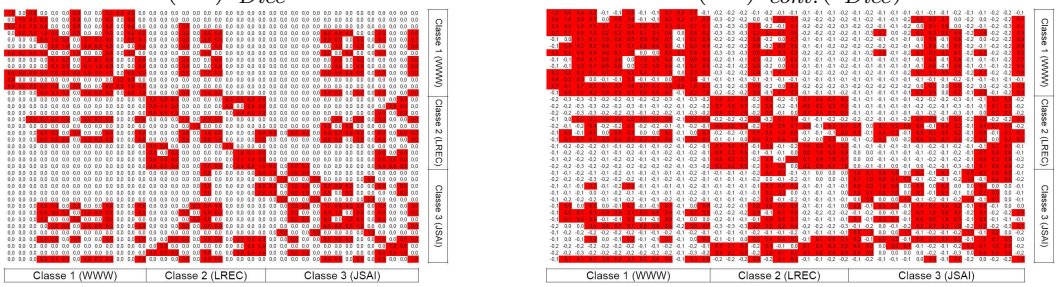
(1.a)IM (1.b)$I_{cont.}(IM)$



(2.a)I_{SDOC} (2.b)$I_{cont.}(I_{SDOC})$



(3.a)I_{Dice} (3.b)$I_{cont.}(I_{Dice})$



(4.a)J (4.b)$I_{cont.}(J)$

FIG. 5.1 – Matrices de proximité obtenues sur les 38 mots-clés : (a) indices de proximité basés sur les cooccurrences, (b) indices de proximité contextuelle.

des matrices indicées (a), et aux coefficients positifs pour les matrices (b). Une mesure pertinente devrait alors permettre de visualiser les trois blocs de cellules grisées, associés aux trois domaines.

On observe tout d'abord que les matrices de droite (indicées b.) permettent de mieux dégager les trois blocs, comparativement aux matrices de gauche (indicées a.), en particulier pour le premier bloc (LREC). Ceci valide de manière globale, l'indice $I_{cont.}$ que nous proposons. Il semble en effet, que l'analyse des vecteurs contextuels produise un indice plus pertinent que l'observation des cooccurrences uniquement.

Les paires de matrices 2, 3 et 4 sont, visuellement assez semblables ; un bloc correspondant aux mots du domaine (LREC) est retrouvé, alors que les deux autres domaines apparaissent moins distinctement. La matrice de proximité contextuelle associée à la mesure d'information mutuelle (matrice 1.b), est incontestablement la plus révélatrice de l'organisation des mots en trois domaines ; ces trois domaines sont distinctement observés, et on note que certains mots semblent communs à plusieurs domaines. Cette dernière observation entretient l'idée que l'organisation de données en groupes non-disjoints est particulièrement adaptée au traitement d'objets textuels, et notamment au traitement des mots.

Après avoir proposé puis validé empiriquement un indice de proximité contextuelle entre mots, nous choisissons d'utiliser cet indice pour la tâche d'organisation des mots en classes contextuelles. Dans ce qui suit, nous utiliserons le jeu de données test établi, ainsi que l'indice $I_{cont.}$ associé à la mesure d'information mutuelle, pour évaluer l'algorithme de regroupement PoBOC.

5.3 Classification de mots

Dans cette section, nous analysons les caractéristiques le l'algorithme PoBOC dans l'application à la construction de classes sémantiques de mots. Dans un premier temps, nous rappelons les principaux travaux antérieurs pour cette tâche ; nous expliquons ensuite les raisons pour lesquelles le type de schéma renvoyé par PoBOC est intéressant dans ce contexte ; enfin, nous proposons des résultats obtenus sur le jeu de données présenté dans la section précédente.

5.3.1 Travaux antérieurs

Il existe une quantité importante de travaux traitant du regroupement contextuel/ sémantique de mots. Ces méthodes sont présentées parfois de manières différentes, par exemple [179] parle d'extraction de *concepts* tandis que [89] identifie les *thèmes* contenus dans des documents scientifiques, enfin [175] utilise une terminologie plus proche de la notre en générant des *classes sémantiques*. Il serait inexacte d'affirmer que toutes ces dénominations désignent un même objectif final, cependant il s'agit, à chaque fois, de construire des groupes de termes/mots, partageant des caractéristiques sémantiques ou syntaxiques. Chacune de ces méthodes fait appel, d'une manière ou d'une autre, à un processus de regroupement, dont les caractéristiques doivent correspondre à la nature des données, autrement dit, les mots.

Les méthodes de regroupement utilisées dans la plupart de ces travaux sont spécifiques à l'application concernée et sont difficilement réutilisables ou généralisables pour d'autres corpus ou d'autres connaissances sur le domaine. Par exemple, les classes sémantiques

générées par [175], sont obtenues par l'analyse de la structure du thésaurus Roget, de même que [2] utilise l'algorithme Cobweb à partir de descriptions syntaxiques et sémantiques des mots, en se basant sur le thésaurus WordNet. Enfin, [53] recherche les schémas syntaxiques récurrents pour identifier les associations entre mots (cette approche est également utilisée dans le système ASIUM [59]). La réutilisation de ces systèmes nécessite donc, soit la donnée d'une base de connaissance, ce qui est difficile à obtenir pour des domaines de spécialité, soit une analyse syntaxique fine, ce qui peut devenir coûteux dans le cas de larges corpus.

D'autres approches, plus générales, commencent par définir les relations entre les mots, à l'aide de poids ou de mesures de (dis)similarité. Ensuite, le processus de clustering peut être considéré indépendamment. Par exemple, [89] considère les thèmes comme des composantes connexes dans un réseau de termes, dont les liaisons sont pondérées relativement aux caractéristiques morphologiques de ces termes. Malheureusement, par définition même d'une composante connexe, les thèmes obtenus restent disjoints. D'autres algorithmes, proposés dans les systèmes GALEX (*Graph Analyzer for LEXicometry*) [179], UNICON (*UNsupervised Induction of Concepts*) [114] ou encore CBC (*Clustering By Committees*) [138] proposent des processus de regroupement assez proches de celui utilisé dans PoBOC, en recherchant un ensemble de "termes pôles" (GALEX) ou "committees" (CBC) auxquels sont affectés ensuite les mots non-traités. La méthodes des k-moyennes axiales [110], présentée dans le chapitre 1, est également utilisée dans ce contexte. Toutes ces méthodes sont orientées vers la construction de classes non-disjointes, cependant, selon le cas, elle nécessitent la détermination *a priori* du nombre de classes ou d'un seuil d'affectation général, et ne tiennent pas compte de la notion de densité pour constituer ces classes.

Ces dernières approches récentes, témoignent de l'intérêt croissant dans les communautés TAL et RI, pour les méthodes de clustering produisant des schémas non-disjoints. Malgré tous ces efforts, et probablement à cause des limites que nous venons de citer, l'utilisation de méthodes simples et reconnues dans le domaine de l'apprentissage non-supervisé mais faiblement performantes pour l'application (comme l'algorithme des k-moyennes), est un reflexe difficile à atténuer [166, 143].

5.3.2 PoBOC : un algorithme approprié ?

L'algorithme de regroupement PoBOC présente de bonnes propriétés pour cette application. Tout d'abord, si l'on s'intéresse à la description des objets, il n'existe pas d'attributs, en nombre raisonnable, permettant de décrire des mots. Le plus souvent, les mots sont caractérisés par l'ensemble des relations qu'ils entretiennent les uns avec les autres. C'est ce que nous avons appelé, dans le chapitre 1, les données relationnelles. Ainsi, l'algorithme de regroupement doit s'appuyer sur cet ensemble de relations, par exemple une matrice de (dis)similarités, pour construire la classification. Cette "contrainte" exclut l'utilisation de certaines méthodes de clustering basées sur une représentation spatiale des données, telles que l'algorithme des k-moyennes[10], les méthodes par découpage de l'espace en grille, etc.

La recherche d'un nombre approprié de groupes est une autre caractéristique de PoBOC. Il est effectivement fréquent d'utiliser le processus de clustering dans la perspective de découvrir des connaissances dans les données. Pour cette application, on souhaitera, le plus souvent, découvrir les thématiques présentes dans un document ou dans un corpus de documents, sans connaître *a priori* le nombre de thèmes abordés.

Poursuivons avec une autre spécificité de PoBOC, qui est la construction de groupes de densités variées. Le voisinage d'un mot, étant donnée le type de mesure de proximité

[10] Il faudra plutôt utiliser une variante ; k-médoïdes par exemple.

que nous avons défini en section précédente, correspond à un ensemble de mots qui lui sont sémantiquement proches. Un groupe dense correspondra donc à un ensemble de mots utilisés dans des contextes très similaires (*e.g.* des mots spécifiques appartenant à un même domaine). Inversement, un groupe de faible densité correspondra à un ensemble de mots dont les contextes sont globalement similaires (*e.g.* des mots plus généraux d'un même domaine ou de plusieurs domaines sémantiquement proches). L'algorithme PoBOC présente alors les dispositions pour extraire à la fois les concepts spécifiques à un domaine et des thématiques abordées de façon plus générale.

Enfin et surtout, la méthode que nous avons proposé autorise un objet, en l'occurrence ici un mot, à appartenir à plusieurs groupes. Nous avons noté dans la section précédente, qu'un même mot peut être utilisé dans plusieurs contextes différents. Sur les deux concepts "gastronomie" et "justice", *avocat* est un exemple typique de mot appartenant au champ lexical de chacun des deux concepts sémantiques. Dans l'évaluation portant sur les trois domaines associés aux conférences et journaux scientifiques, nous avons observé que certains mots présentaient des coefficients de proximité positifs avec des mots de domaines différents. L'organisation de ces mots en groupes disjoints reviendrait alors à ne reconnaître qu'un seul contexte d'utilisation pour chaque mot. De même, il est intéressant de définir des groupes "durs"[11], afin de conserver la contrainte liée à la simplification de l'ensemble de données. En d'autres termes, il est important de pouvoir conserver une décision binaire sur le fait que deux mots sont ou non en relation (dans un même groupe).

Nous venons d'énumérer un ensemble de caractéristiques qui laissent penser que l'algorithme PoBOC est adapté à la tâche d'apprentissage de classes contextuelles à partir d'un ensemble de mots. Il nous reste à présent à confirmer ces intuitions sur une situation réelle.

5.3.3 Expérimentation

Nous utilisons le jeu de données constitué des 38 mots-clés issus des articles scientifiques de conférences et journaux internationaux. Rappelons que nous avions construit, sur cette base, une matrice de proximités contextuelles entre les mots, et que c'est cette matrice qui est passée en paramètre de l'algorithme PoBOC.

La figure 5.2 présente le pseudo-partitionnement obtenu, sur ce jeu de données, avec l'algorithme PoBOC. On observe alors 7 clusters non-disjoints, que nous avons organisés de manière hiérarchique[12] afin de mieux appréhender les proximités entre ces différents groupes. Les mots apparaissant dans chacun des clusters sont étiquetés par le domaine dont ils sont issus (LREC, JSAI et WWW pour les domaines *Ressources du Langage*, *Intelligence Artificielle* et *Web Mondial* respectivement). Pour plus de lisibilité, les mots se trouvant à l'intersection de plusieurs clusters sont dupliqués dans chacun de ces clusters.

On remarque en premier lieu que les 7 groupes obtenus sont plutôt homogènes, une étiquette étant systématiquement majoritaire dans chacun de ces groupes. De plus, les trois groupes obtenus au sommet de la hiérarchie correspondent assez fidèlement aux 3 domaines attendus :

- $C_{1.1}$, $C_{1.2} \cup C_{1.3}$ sont fusionnés dans C_1, et chacun de ces trois clusters contient majoritairement les mots du domaine WWW ; le concept du Web Mondial est donc retrouvé

[11] Nous opposons à groupes "durs", les groupes "flous" définis par un ensemble de valeurs d'appartenance des objets au groupe.

[12] L'arbre hiérarchique est obtenu par l'algorithme agglomératif hiérarchique du lien simple.

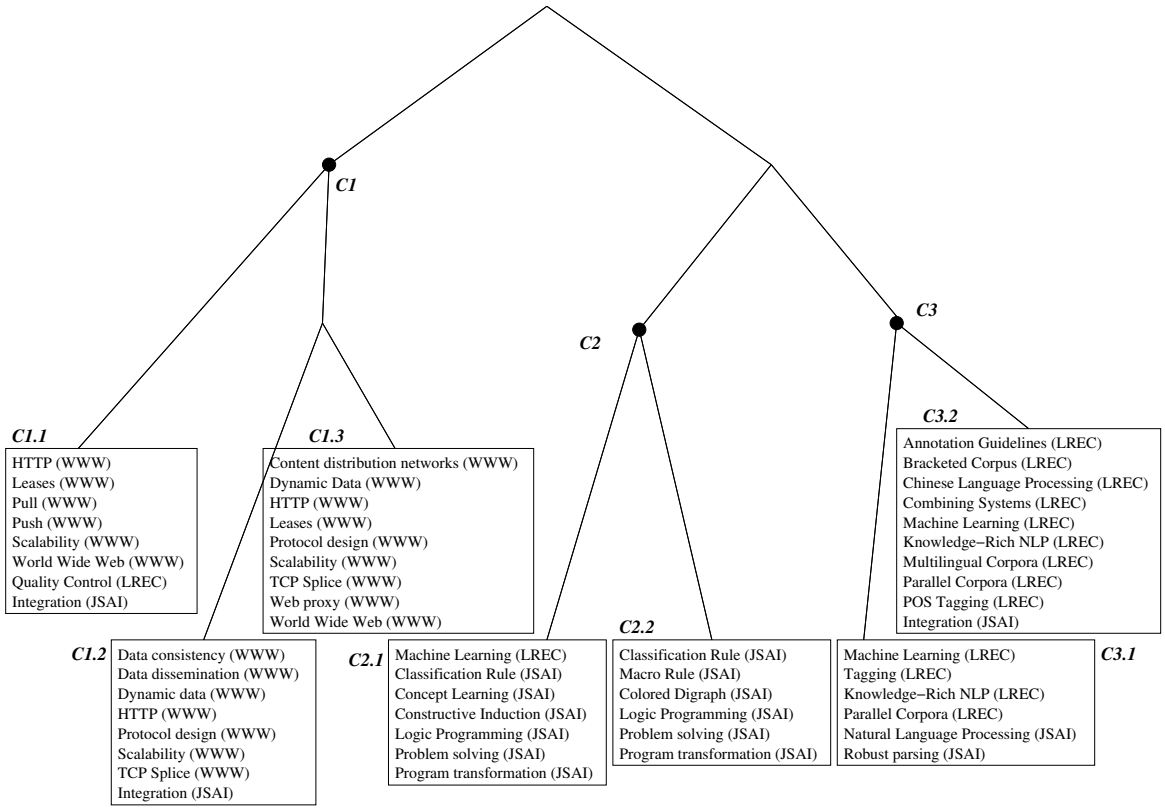


FIG. 5.2 – Organisation hiérarchique du pseudo-partitionnement obtenu par PoBOC sur la base des 38 mots-clés.

dans le nœud C_1,

- $C_{2.1}$, $C_{2.2}$ sont fusionnés dans C_2, et chacun de ces deux clusters contient majoritairement les mots du domaine JSAI ; le concept de l'Intelligence Artificielle est donc retrouvé dans le nœud C_2,
- $C_{3.1}$, $C_{3.2}$ sont fusionnés dans C_3, et chacun de ces deux clusters contient majoritairement les mots du domaine LREC ; le concept nommé Ressources du Langage est donc retrouvé dans le nœud C_3.

Dans un deuxième temps, en observant de façon plus précise les groupes obtenus, on note, par rapport à la classification de référence, les modifications suivantes :

– le mot *Machine Learning*, extrait d'un articles scientifique de la conférence LREC, se retrouve entre autre dans le cluster $C_{2.1}$, contenant 6 autres mots, tous étiquetés JSAI (cette remarque est encore amplifiée au niveau du groupe C_2),

– les mots *Natural Language Processing* et *Robust Parsing*, tous deux extraits de communications publiées dans le journal JSAI apparaissent dans des groupes de mots ($C_{3.1}$ et C_3) majoritairement étiquetés LREC.

Ces modifications, qui sont comptabilisées comme des erreurs lorsque l'on compare le schéma généré par PoBOC (noté $\mathcal{P}$) avec la classification de référence (notée $\mathcal{C}$), doivent plutôt être interprétées comme une correction de $\mathcal{C}$. En effet, le mot *Machine Learning* par exemple, est rarement utilisé comme mot-clé pour un article présenté dans un journal

dont la thématique est l'Intelligence Artificielle[13]. Il est cependant naturel de le mettre en relation avec les mots *Classification Rule*, *Concept Learning* ou encore *Logic Programming*. Cette remarque est valable pour les deux autres mots *Natural Language Processing* et *Robust Parsing*, qui retrouvent leur place naturelle dans $\mathcal{P}$.

Intéressons nous, à présent, aux intersections observées entre les 7 groupes générés par PoBOC. Notons pour commencer que le coefficient de partition est nettement plus élevé pour la pseudo-partition initiale $\mathcal{P} = \{C_{1.1}, C_{1.2}, C_{1.3}, C_{2.1}, C_{2.2}, C_{3.1}, C_{3.1}\}$ (PC($\mathcal{P}$)=1.18) que pour la pseudo-partition $\mathcal{P}' = \{C_1, C_2, C_3\}$ obtenue après hiérarchisation (PC($\mathcal{P}'$)=1.03). La plupart des intersections concernent, en effet, des groupes d'un même domaine. On distingue deux catégories de mots pouvant donner lieu à des intersections entre clusters :

- Les mots n'appartenant pas à un vocabulaire de spécialité, comme par exemple le mot *integration*, qui peut être utilisé dans plusieurs contextes. Ce mot se retrouve donc dans des clusters très différents tels que $C_{1.1}$, $C_{1.2}$ et $C_{3.2}$.
- Les mots spécifiques et représentatifs d'un domaine. Les mots *HTTP*, *Logic Programming* ou encore *Knowledge Rich NLP* sont spécifiques à chacun des trois domaines WWW, JSAI et LREC respectivement et apparaissent dans tous les groupes concernés ; *HTTP* appartient aux trois groupes $\{C_{1,i}\}_{i=1,2,3}$, *Logic Programming* aux deux groupes $\{C_{2,i}\}_{i=1,2}$ et *Knowledge Rich NLP* aux deux groupes $\{C_{3,i}\}_{i=1,2}$.

Pour compléter l'étude qualitative précédente, nous proposons une comparaison avec d'autres schémas obtenus par des algorithmes traditionnels hiérarchiques (algorithmes agglomératifs hiérarchiques du lien moyen (ALink), complet (CLink) et simple (SLink)) et par partitionnement (k-médoïdes). Ces schémas sont évalués à l'aide des indices de qualités déjà utilisés dans le chapitre 2, à savoir :

- La *statistique de Huberts*, permettant de mesurer l'adéquation entre la partition obtenue et la partition de référence ($\Gamma_{externe}$), ou entre la partition obtenue et la matrice de proximité utilisée ($\Gamma_{interne}$).
- Les mesures d'inerties, donnant des indications sur la cohésion interne des groupes (inertie intra-cluster), et la séparation des groupes (inertie inter-clusters).

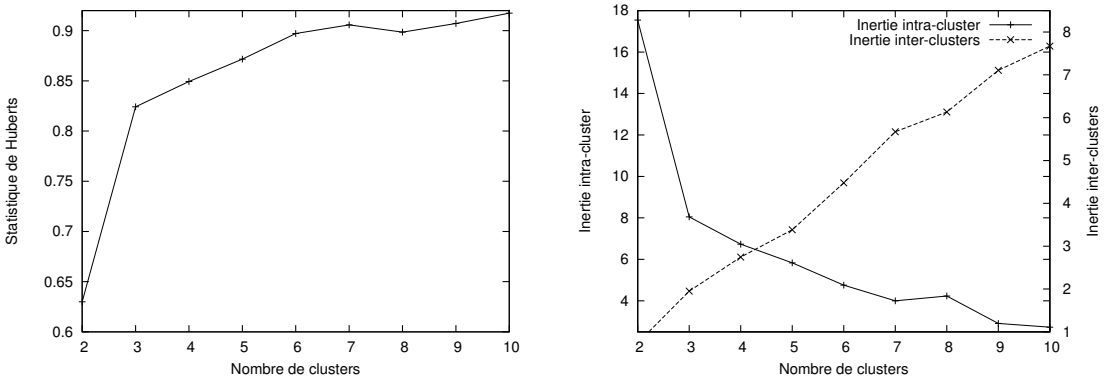


FIG. 5.3 – Évaluation du nombre de clusters sur les 38 mots-clés.

Le tableau 5.6 rend compte des évaluations effectuées sur l'ensemble des 38 mots-clés, en comparant les schémas $\mathcal{P}$ et $\mathcal{P}'$ obtenus par l'algorithme PoBOC, avec ceux générés

[13]Ce mot serait trop général pour ce domaine et apporterait peu de précision sur le contenu de l'article. En revanche, il apporte davantage d'informations lorsqu'il est utilisé dans une communication d'un autre domaine, par exemple pour une conférence du type LREC.

Algorithme	Nb. clusters	$\Gamma_{interne}$ ↗	$\Gamma_{externe}$ ↗	Inertie intra-cluster ↘	Inertie inter-clusters ↗
PoBOC	7	0.88	**0.80**	5.79	4.54
k-médoïdes		0.88	0.75	3.86	5.14
ALink		0.87	0.76	3.33	5.17
CLink		0.88	0.75	4.90	5.82
SLink		0.86	0.75	5.33	4.79
PoBOC	3	0.82	**0.74**	9.92	2.03
k-médoïdes		0.82	0.74	8.34	2.24
ALink		0.83	0.71	8.84	2.24
CLink		0.79	0.73	9.72	2.24
SLink		0.16	0.14	28.05	1.73

TAB. 5.6 – Évaluation quantitative des schémas obtenus par différentes méthodes de clustering.

par les méthodes de regroupement précitées, et paramétrées avec un nombre de clusters $k = 7$ (nombre de clusters proposé par PoBOC) puis $k = 3$ (nombre de groupes dans la classification de référence).

On observe que les inerties calculées sur les schémas générés par PoBOC sont médiocres, comparativement aux clusters compacts et séparés obtenus par k-médoïdes ou ALink par exemple. Comme nous l'avons déjà évoqué lors d'une précédente analyse, ce résultat est dû, en grande partie, au fait que ces mesures d'inerties pénalisent, par définition, les schémas constitués de groupes non-disjoints.

En revanche, sur les deux indices indépendants du type de schémas (indices Γ), PoBOC obtient les meilleurs résultats de clustering, en particulier pour la configuration en 7 groupes, initialement proposée. L'étape de hiérarchisation induit une dégradation prévisible, mais qui maintient PoBOC au rang des meilleures méthodes (contrairement à la méthode SLink notamment).

Enfin, les tracés présentés en figure 5.3 indiquent l'évolution des indices relatifs (cf. évaluation de PoBOC, chapitre 2), lorsque le paramètre k (nombre de clusters) augmente. Ces résultats sont obtenus en effectuant une moyenne sur 10,000 schémas obtenus par l'algorithme des k-médoïdes.

On observe sur ces tracés, que les paramétrages $k = 3$ et $k = 7$ correspondent effectivement à deux configurations pertinentes. Cela ce manifeste par des "cassures" ou "coudes" communs sur au moins deux des trois indices.

Dans la suite de ce chapitre, nous utilisons le processus de regroupement de PoBOC dans une tâche plus complexe de RI. Nous avons montré, dans le système RuLe-Clust, l'intérêt d'organiser des exemples en groupes non-disjoints avant de lancer le processus d'apprentissage de règles ; dans la même idée, nous proposons, dans la section suivante, de regrouper les mots extraits d'un corpus de documents, avant d'entreprendre le processus d'apprentissage pour la classification automatique de documents.

Les groupes de mots obtenus permettent en effet de mieux décrire les documents (description plus concise voire plus précise) ; on peut parler également d'indexation de ces documents, par des groupes de mots, plutôt que par les mots eux-mêmes.

5.4 Regroupement de mots pour la classification de documents

5.4.1 Introduction à la classification de documents

La classification automatique de documents est un problème important à l'intersection de deux domaines de recherche majeurs : l'Apprentissage Automatique et la Recherche d'Information. On distingue la classification supervisée de la classification non-supervisée, selon que des exemples de documents étiquetés sont disponibles ou non pour l'apprentissage.

Le regroupement (non-supervisé) de documents est utilisé en Recherche d'Information sous l'hypothèse que des documents de contenus similaires sont pertinents pour les mêmes requêtes [181]. Le processus de classification non-supervisée peut intervenir dans différentes étapes d'un système de recherche d'information. Par exemple, on peut choisir d'organiser *a priori* la collection de documents afin d'en accélérer l'interrogation, ou encore d'organiser les documents fournis en réponse à une requête dans le but de simplifier et d'assister l'utilisateur dans sa recherche[14].

Etant donné un modèle de classification, le problème de classification supervisée de documents[15], consiste à apprendre les paramètres de ce modèle à partir des observations issues d'un corpus d'entraînement, constitué de documents étiquetés (document, classe). Ce modèle paramétré permet alors de prédire la classe d'appartenance d'un nouveau document test. La classification (supervisée) de documents trouve ses principales applications en recherche d'information : l'indexation automatique, le routage ou filtrage d'information en temps-réel (mail, notes de service, spam, etc.) ou encore la désambiguïsation sémantique [164].

Dans ce qui suit, nous nous situons dans cette dernière approche de classification supervisée de documents. Nous en précisons alors les principales problématiques en donnant un aperçu des différentes stratégies proposées dans la littérature.

5.4.2 Classification supervisée de documents

L'élaboration d'un classifieur pour la tâche de classification automatique de documents, nécessite un processus en trois étapes :

1. l'extraction d'attributs pertinents,
2. la phase d'apprentissage à partir du corpus d'entraînement,
3. l'évaluation du classifieur.

Ces trois étapes, que nous précisons par la suite, constituent chacune un sous-domaine de recherche à part entière [1].

L'extraction d'attributs pertinents

C'est l'étape du processus que nous visons dans cette étude. Vu que l'on cherche à traiter des documents relativement à leur contenu (thématique), il est naturel de considérer, comme descripteur de ce contenu, l'unité de base d'un document, à savoir le mot. Ainsi, dans les systèmes de classification automatique de documents, utilisés dans un contexte

[14] Ce type de procédé est utilisé, par exemple, dans le moteur de recherche *www.vivisimo.com*.
[15] En anglais *document categorization*.

de recherche d'information, chaque document est généralement représenté par un "sac de mots". Il est évident que beaucoup de mots apparaîssant dans un document ne décrivent pas son contenu (*e.g. le, a, est, pour, que,* etc.), ces mots sont appelés communément "mots vides". L'extraction des mots pertinents (ou porteurs de sens) constitue alors un premier traitement indispensable et généralement réalisé à l'aide de mesures de fréquences [159] (*term frequency, tf×idf, tfc,* etc.). Ces mesures se fondent sur les deux principes suivants :

- Plus un mot est fréquent dans un document, plus il est pertinent pour décrire la thématique de ce document.
- Plus un mot est fréquent dans l'ensemble des documents d'une collection, moins il est pertinent pour discriminer les documents entre eux.

On estime que le filtrage, décrit précédemment, revient à réduire de 40 à 50% le vocabulaire utilisé dans une collection de documents. Malgré tout, l'ensemble des descripteurs reste très grand (souvent plusieurs milliers de mots), ce qui constitue un inconvénient majeur pour les techniques standard de classification (*e.g.* arbres de décision, algorithmes génétiques, etc.). Ainsi, à partir du constat que la matrice *documents×mots* est très éparse (beaucoup de 0), différentes stratégies complémentaires ont été envisagées pour réduire la dimension de l'espace de représentation des documents. Les trois principales stratégies utilisées sont :

- La sélection d'attributs : par l'utilisation de mesures de gain d'information ou tests statistiques,
- Le reparamétrage : par construction de nouveaux attributs obtenus par combinaisons et/ou transformations des attributs initiaux,
- Le regroupement : par utilisation de techniques de clustering pour générer des groupes d'attributs "corrélés".

Ces trois principes seront précisés dans la section suivante, en nous intéressant plus particulièrement au troisième principe, faisant appel à des méthodes de classification non-supervisées, proches de nos préoccupations.

Les différents classifieurs

Il existe deux types de classifications : automatique ou semi-automatique. On parle de classification *automatique* lorsque, étant donnés un document d et une classe c, le classifieur renvoie une décision binaire sur l'appartenance de d à la classe c ($d \in c$ ou $d \notin c$). En revanche, une classification semi-automatique revient à apprendre, pour un document d, une liste de scores correspondant, en quelque sorte, aux indices de confiance des faits $\{d \in c_i\}$. Dans les deux cas, on peut résumer le problème de classification à l'apprentissage des fonctions CSV_i (*categorization status value*) telles que, pour une classe cible c_i :

$CSV_i : \mathcal{D} \rightarrow \{Vrai, Faux\}$ (classification automatique),
$CSV_i : \mathcal{D} \rightarrow [0, 1]$ (classification semi-automatique).

Parmi les méthodes de classification automatique de documents, on peut citer, entre autre, les arbres de décision avec l'utilisation, par exemple, de l'algorithme C4.5 [32], les règles de décision avec le système CHARADE [130] ou encore les Machines à Support Vectoriel[16] (SVM) [93].

[16] Également appelées "Séparateurs à Vaste Marge".

En ce qui concerne les systèmes de classification semi-automatiques, des fonctions CSV de différents types peuvent être apprises :

- Les fonctions *probabilistes* apprises, par exemple, à l'aide du classifieur naïf de Bayes [156]. Les valeurs obtenues correspondent aux probabilités *a priori* qu'un document appartienne à une classe donnée.
- Les fonctions basées sur des *distances*, en utilisant la méthode Rocchio [87] ou d'autres approches du type : classification par "plus proches voisins". Dans ce cas, la fonction CSV est relative à la "distance" entre un document test, et le(s) représentant(s) de la classe cible.
- Les fonctions de *poids* issues de méthodes de régression linéaire [188] ou de réseaux de neurones [33]. Dans le premier cas, le problème consiste à apprendre, pour un document d, un vecteur de poids dans lequel chaque composante est associée à une classe, à partir des vecteurs binaires issus des documents d'entraînement. Dans la seconde méthode, les mots contenus dans le document d à classer, sont "propagés" dans un réseau pour lequel chaque mot représente un neurone d'entrée et chaque classe, un neurone de sortie. Abusivement, on peut parler de "propagation" du document dans le réseau, suite à laquelle, les valeurs obtenues sur les neurones de sortie déterminent les décisions du classifieurs.

Dans ce tavail, nous étudierons la classification "automatique" des documents par le classifieur naïf de Bayes. Pour simplifier, nous poserons la convention suivante : étant donnés un document test d, et une classe cible c_k, d est classé dans c_k si et seulement si $CVS_i(d)$ est maximale pour $i = k$. Un document est donc classé dans une seule classe, correspondant à celle dont la probabilité d'appartenance *a priori* de d est la plus élevée.

Le problème de l'évaluation

L'existence de nombreuses mesures d'évaluation pour les classifieurs est révélatrice de la difficulté de cette tâche. En effet, différents cas de figures sont envisageables :

Classification binaire simple : On considère que chaque document est étiqueté par une seule classe, et que le classifieur décide de "la" classe d'appartenance. Dans ce cas, la mesure traditionnelle de *justesse*[17], déjà utilisée dans le chapitre 3, permet d'évaluer simplement, l'efficacité du classifieur. Rappelons que cette mesure est définie par

$$J = \frac{vp + vn}{vp + vn + fp + fn}$$

où les notations vp, vn, fp et fn désignent respectivement le nombre de "vrais positifs", de "vrais négatifs", de "faux positifs" et de "faux négatifs", définis dans les tables de contingence présentées dans les tableaux 5.7. C'est dans cette situation de classification binaire simple que se placeront nos expérimentations (section 5.4.8).

Classification binaire multiple : Chaque document d'entraînement est étiqueté par une ou plusieurs classes, et le classifieur peut décider de l'appartenance d'un document test à une ou plusieurs classes. On fait alors appel aux mesures de *précision* et de *rappel*, traditionnelles en RI. Pour une classe c_i donnée, la précision (P_i) mesure la proportion de documents bien classés parmi ceux classés dans c_i, tandis que le

[17] Nous utilisons ici le terme *"justesse"*(en anglais *"accuracy"*), pour éviter la confusion avec le terme *"précision"* qui sera utilisé par la suite pour désigner une autre mesure.

Appartenance à la Classe c_i		Jugement expert	
		OUI	**NON**
Jugement classifieur	**OUI**	vp_i	fp_i
	NON	fn_i	vn_i

Appartenance aux classes $\{c_i\}_{i=1\ldots m}$		Jugement expert	
		OUI	**NON**
Jugement classifieur	**OUI**	$vp = \sum_{i=1}^{m} vp_i$	$fp = \sum_{i=1}^{m} fp_i$
	NON	$fn = \sum_{i=1}^{m} fn_i$	$vn = \sum_{i=1}^{m} vn_i$

TAB. 5.7 – Tables de contingence pour la classe c_i (gauche) et globale (droite).

rappel (R_i) mesure la proportion de documents classés dans c_i, parmi ceux qui sont effectivement étiquetés c_i.

$$P_i = \frac{vp_i}{vp_i + fp_i} \quad ; \quad R_i = \frac{vp_i}{vp_i + fn_i}$$

La précision et le rappel peuvent ensuite être évalués sur l'ensemble des classes $c_1, \ldots, c_m$ par l'une ou l'autre des méthodes suivantes :

- *micro-précision/rappel* :

$$\tilde{P}^{\mu} = \frac{vp}{vp + fp} \quad ; \quad \tilde{R}^{\mu} = \frac{vp}{vp + fn}$$

- *macro-précision/rappel* :

$$\tilde{P}^{M} = \frac{\sum_{i=1}^{m} P_i}{m} \quad ; \quad \tilde{R}^{M} = \frac{\sum_{i=1}^{m} R_i}{m}$$

D'autres critères combinant les notions de précision et de rappel sont proposés ; il s'agit par exemple de la recherche du point d'équilibre (*Break-even point*) [6] ou de la F_β-mesure [181].

Classification semi-automatique : Chaque document est étiqueté par une ou plusieurs classes et le classifieur renvoie, pour un document test donné, une liste de scores. Pour tenir compte de ces valeurs, [189] propose la *mesure de précision moyenne sur 11 points* (*Eleven-point average precision*). Cette mesure considère 11 "seuils" $(0.0, 0.1, \ldots, 1.0)$ et calcule pour chacun de ces seuils, la précision et le rappel des classifications binaires, issues des listes de scores seuillées.

5.4.3 Réduction de la dimension de l'espace de description des documents

Par sélection d'un sous-ensemble d'attributs

Étant donné un ensemble d'attributs (ou descripteurs) V, la réduction de la dimension de l'espace de description, peut être effectuée par sélection d'un sous-ensemble $V' \subset V$. On recherche alors le plus petit sous-espace V', contenant l'information suffisante sur les documents, pour aboutir à un classifieur au moins aussi performant dans V' que dans V. La recherche de ce sous-ensemble optimal V' est effectuée sur les données d'entraînement ; il s'agit d'un problème NP-complet (l'espace des hypothèses correspond à l'ensemble des parties $\mathcal{P}(V)$). Cette recherche est donc généralement guidée par des mesures de pertinence sur les attributs de V (ici les mots du vocabulaire).

Le *seuil de fréquence de documents* [189], par exemple, est fondé sur l'hypothèse que les mots rares (apparaîssant dans peu de documents) sont peu utiles ou peu influents dans la prédiction des catégories (classes de documents). Il s'agit alors de calculer, pour chaque mot w de V, le rapport $N_w(D)/|D|$, où D désigne le corpus d'entraînement et $N_w(D)$, le nombre de documents contenant w dans D. Seuls les mots apparaîssant dans le plus grand nombre de documents sont retenus. Notons que l'utilisation de ce critère, suppose une étape préalable de suppression des mots non pertinents (mots vides) comme nous l'avons précisé dans la section précédente.

La *mesure du* χ^2 [161], évalue la dépendance entre un mot w et une classe de documents c_i. Pour cela, le calcul suivant compare les proportions de documents contenant w ou ne contenant pas w ($\overline{w}$), dans la classe c_i ou dans les documents des autres classes ($\overline{c_i}$) :

$$\chi^2(w, c_i) = \frac{|D|.[p(w, c_i).p(\overline{w}, \overline{c_i}) - p(w, \overline{c_i}).p(\overline{w}, c_i)]^2}{p(w).p(\overline{w}).p(c_i).p(\overline{c_i})}$$

Dans cette définition, $p(\overline{w}, c_i)$, par exemple, désigne la probabilité que, pour un document aléatoire d, le mot w n'occurre pas dans d et d appartient à la classe c_i. Ces probabilités sont estimées sur le corpus d'entraînement D. Deux configurations sont ensuite possibles pour évaluer l'indépendance de w par rapport à l'ensemble des classes :

$$\chi^2(w) = \sum_{i=1}^{m} p(c_i).\chi^2(w, c_i) \qquad \text{(moyenne pondérée)}$$

$$\chi^2_{max}(w) = \max_i \chi^2(w, c_i) \qquad \text{(maximum)}$$

Les mots selectionnés sont ceux qui sont les moins indépendants des classes (valeurs élevées pour ces mesures).

De nombreuses autres mesures utilisent des comparaisons proches de celles présentées dans la mesure du χ^2. On peut noter par exemple, le *facteur d'association DIA* [62], le *gain d'information* [111] ou encore l'*information mutuelle* [105]. Sur les expérimentations effectuées, ces mesures permettent de réduire la dimension de l'espace jusqu'à un facteur 100, sans perte, voire avec un léger gain de performance sur la classification.

On oppose aux méthodes de sélection d'attributs, les méthodes d'extraction d'attributs, dont les deux techniques à suivre font partie.

Par reparamétrage de l'espace

Le re-paramétrage de l'espace des attributs consiste à définir un nouvel ensemble d'attributs, chaque nouvel attribut étant une combinaison linéaire des attributs initiaux. Cette méthode, mieux connue sous le nom de *Latent Semantic Indexing* (LSI) [38] dans le cadre de l'application à la classification de documents, est basée sur une décomposition en valeurs propres, de la matrice (*documents*×*mots*). La réduction de la dimension de l'espace est réalisée par sélection des plus grandes valeurs propres.

Cette méthode est reconnue pour être particulièrement adaptée au traitement de certaines particularités liées à la langue naturelle, comme par exemple les synonymes. En effet, deux mots synonymes correspondent à deux dimensions différentes dans l'espace initial alors qu'ils ont un comportement similaire dans le nouvel espace de plus faible dimension. Ainsi, des documents ayant peu de mots en commun peuvent être très similaires dans le nouvel espace, pour peu que des mots synonymes apparaissent dans ces documents.

Finalement, cette méthode est une manière de découvrir la structure sémantique latente (cachée) du vocabulaire.

La capacité de LSI à réduire la dimension de l'espace de description des documents, est supérieure aux approches par sélection d'attributs. Cependant, les nouvelles dimensions extraites sont difficilement interprétables.

Récemment, T. Hofmann proposait une alternative davantage formelle : le modèle génératif pLSI (*probabilistic Latent Semantic Indexing*) [85]. L'approche pLSI considère que chaque mot est généré par un thème et chaque thème est modélisé par une loi multinomiale, dont les paramètres sont approximés par l'algorithme EM. On peut alors considérer qu'un document est généré par un mélange de lois de probabilités (différents thèmes). Ce dernier point est intéréssant car, contrairement à la plupart des approches d'indexation, la méthode pLSI ne fixe pas le document comme unité thématique de base et prend en compte le fait qu'un document puisse faire référence à plusieurs thèmes.

Par regroupement des attributs

Une dernière technique de réduction de l'espace, consiste à regrouper les attributs similaires, de façon à utiliser les groupes de mots plutôt que les mots eux-mêmes comme dimensions de l'espace. Cette technique se place donc dans la continuité de notre chapitre, puisqu'il s'agit de construire des groupes de mots sémantiquement proches.

La méthode de réduction de l'espace par regroupement d'attributs nécessite de définir une mesure de similarité "sémantique" puis de choisir un algorithme de clustering approprié. Lewis [111] est le premier à utiliser des groupes de mots pour classer des documents. En utilisant une mesure de similarité basée sur une connaissance extérieure (WordNet) puis en construisant des paires de mots réciproquement proches, les résultats obtenus sont inférieurs à ceux obtenus sans réduction de l'espace. Une seconde tentative [112], utilisant une mesure de similarité basée sur la notion de cooccurrence dans les documents d'entraînement, couplée d'un algorithme de clustering hiérarchique, n'apportera toujours pas d'amélioration significative.

En revanche, l'utilisation d'un algorithme de "clustering supervisé" permet une réduction très importante de la dimension de l'espace (d'un facteur supérieur à 10,000), en conservant, voire en améliorant, les performances du classifieur. Les systèmes ADC (*Agglomerative Distributional Clustering*) [7] et ITDC (*Information-Theoretic Divisive Clustering*) [42] utilisent tous deux une mesure de similarité basée sur l'analyse des distributions des mots sur les classes de documents, dans le corpus d'entraînement. L'utilisation des étiquettes de classe, à ce stade du processus, justifie le terme de clustering "supervisé". En revanche, ces deux méthodes divergent sur l'algorithme de clustering utilisé ; le système ADC utilise un algorithme agglomératif hiérarchique tandis que ITDC procède par partitionnement. Comme nous l'étudierons dans la suite, ces deux algorithmes sont des adaptations incrémentales d'algorithmes traditionnels. L'incrémentalité est indispensable, dans cette application où le nombre d'objets à traiter est très grand.

Nous avons montré dans les sections précédentes, que l'organisation de mots en groupes non-disjoints convient particulièrement aux données textuelles. Nous avons donc des raisons de penser que ce type de schéma de classification pourrait convenir, voire améliorer l'étape de réduction de l'espace de description des documents.

Nous choisissons de considérer le cadre de la classification par l'algorithme naïf de Bayes, que nous présentons dans la prochaine section. Nous préciserons ensuite les systèmes

ADC et ITDC, sur lesquels nous comparerons notre approche de clustering avec recouvrements. Cette dernière sera présentée immédiatement après. Enfin, nous proposerons une première série d'expérimentations sur les deux corpus *20Newsgroup* [104] et *Reuters-21578*[18], traditionnellement utilisés comme corpus de référence pour l'évaluation de systèmes de classification automatique de documents.

Dans la suite, nous utilisons les notations suivantes : le corpus d'entraînement est noté $D = \{d_1, \ldots, d_n\}$, le vocabulaire extrait de D est noté $V = \{w_1, \ldots, w_l\}$ et $C = \{c_1, \ldots, c_m\}$ correspond aux classes (étiquettes des documents). Enfin, $N(w_t, d_i)$ désigne le nombre d'occurrences du mot w_t dans le document d_i.

5.4.4 Le classifieur naïf de Bayes pour la classification de documents

Classification à partir de mots

Le classifieur naïf de Bayes est connu pour ses performances en classification automatique de documents [156]. Ce classifieur consiste à apprendre un modèle de génération de documents pour chaque classe. Ces modèles sont définis par les distributions $p(d_i|c_j)$ exprimant le probabilité *a priori* que le document d_i soit généré par la classe c_j (5.1).

$$p(d_i|c_j) = p(|d_i|) \prod_{\{w_t \in d_i\}} p(w_t|c_j) \tag{5.1}$$

Cette expression est obtenue sous l'hypothèse d'indépendance entre les mots (notament sur l'ordre d'apparition des mots), correspondant ici à l'hypothèse "naïve" de Bayes. On sait que cette hypothèse n'est pas vérifiée, cependant les études précédentes montrent qu'en pratique, les performances du classifieur naïf de Bayes pour la classification de documents restent bonnes sous cette hypothèse d'indépendance [49]. Dans l'équation (5.1), $p(w_t|c_j)$ est estimée sur le corpus d'entraînement *via* la règle de succession de Laplace (5.2) :

$$p(w_t|c_j) \approx \frac{1 + \sum_{\{d_i \in c_j\}} N(w_t, d_i)}{l + \sum_{\{w_s \in V\}} \sum_{\{d_i \in c_j\}} N(w_s, d_i)} \tag{5.2}$$

Pour classer un nouveau document d, on observe la probabilité *a priori* $p(c_j|d)$. Ainsi, la classe prédite pour le document d sera celle qui maximize cette probabilité, calculée par la règle de Bayes (5.3) :

$$p(c_j|d) = \frac{p(c_j)p(d|c_j)}{p(d)} \tag{5.3}$$

Par étapes succéssives de transformations et simplifications mathématiques[19] sur l'équation (5.3), le classifieur de Bayes peut être reformulé ainsi :

$$c^*(d) = \arg\max_{j=1\ldots m} \quad \frac{\log p(c_j)}{|d|} + \sum_{\{w_t \in V\}} p(w_t|d) \log p(w_t|c_j) \tag{5.4}$$

Dans cette dernière équation, $p(c_j)$ est donnée par le rapport du nombre de documents étiquetés c_j dans D sur le nombre de documents dans D.

[18] http ://www.research.att.com/~lewis/reuters21578.html
[19] Pour plus de détails, voir [42]

Classification à partir de groupes de mots

Dans le cadre du regroupement de mots, on note $\mathcal{W} = \{W_1, \ldots, W_p\}$ le résultat d'un processus de clustering sur le vocabulaire V, tel que chaque cluster W_i est un sous ensemble de V et l'union de tous les clusters de $\mathcal{W}$ est égale à V. Les equations (5.1) et (5.4) doivent être modifiées en remplaçant w par W puisque les documents ne sont plus caractérisés par des mots mais par des groupes de mots. Les expressions $p(W_s|c_j)$ et $p(W_s|d)$ sont définies par les équations suivantes, dans le cas classique d'un regroupement en clusters disjoints (on dit aussi que $\mathcal{W}$ est une partition stricte de V) :

$$p(W_s|c_j) = \frac{\sum\limits_{\{d_i \in c_j\}} N(W_s, d_i)}{\sum\limits_{\{W_k \in \mathcal{W}\}} \sum\limits_{\{d_i \in c_j\}} N(W_k, d_i)} \quad ; \quad p(W_s|d) = \frac{N(W_s, d)}{|d|} \tag{5.5}$$

où $N(W_k, d) = \sum_{\{w_t \in W_k\}} N(w_t, d)$. Dans notre étude ce modèle doit être adapté au cas de clusters non-disjoints (ou pseudo partition). Nous présentons cette adaptation en section 5.4.7.

5.4.5 Regroupement distributionnel agglomératif

La méthode ADC (*Agglomerative Distributional Clustering*) a été proposée en 1998 par Baker et McCallum [7]. Ce travail est basé sur la théorie distributionnelle initiée en 1993 par Pereira et al. [142]. Chaque mot w_t de V est alors caractérisé par :

- sa distribution sur les classes $p(C|w_t) = \{p(c_j|w_t)\}_{j=1\ldots m}$,
- sa probabilité d'apparition $p(w_t)$.

Ces caractéristiques sont estimées à partir du corpus d'entraînement.

$$p(c_j|w_t) = \frac{\sum_{\{d_i \in c_j\}} \delta(w_t, d_i)}{\sum_{\{d_i \in \mathcal{D}\}} \delta(w_t, d_i)} \tag{5.6}$$

où $\delta(w_t, d_i)$ vaut 1 si w_t est présent dans d_i et 0 sinon.

Le processus de clustering a pour objectif de constituer des groupes de mots, jouant des rôles "similaires" dans le processus de classification. Plutôt que de définir la similarité relativement aux cooccurrences des mots dans les documents, les auteurs s'intéressent alors à leurs cooccurrences dans les classes de documents. La mesure de similarité doit donc tenir compte des étiquettes de classe, c'est en ce sens que le processus de clustering est qualifié de "clustering supervisé".

Étant données deux distributions $p(C|w_t)$ et $p(C|w_s)$, caractéristiques des mots w_t et w_s, la divergence de Kullback-Leibler (KL), peut être utilisée comme indice de comparaison (5.7).

$$D_{KL}(p(C|w_t)||p(C|w_s)) = \sum_{j=1}^{m} p(c_j|w_t) \log \frac{p(c_j|w_t)}{p(c_j|w_s)} \tag{5.7}$$

Cependant cette mesure n'est pas symétrique et n'est pas définie dans le cas où la probabilité $p(c_j|w_s)$ est nulle. En pratique on utilise alors plutôt la "divergence de KL à la moyenne" (5.8) :

Algorithme ADC : *Agglomerative Distributional Clustering*

Entrées : Le vocabulaire V, caractérisé par :
-les distributions $\{p(C|w_t)\}_{w_t \in V}$,
-les probabilités $\{p(w_t)\}_{w_t \in V}$,
l le nombre de classes de documents,
k le nombre de clusters désiré.

Sortie : Une partition $\mathcal{W} = \{W_1, \ldots, W_k\}$ de V en k clusters disjoints.

1. Ordonner V relativement à l'information mutuelle des mots avec la variable de classe $I(w; C)$

2. Initialiser k singletons $W_1, \ldots, W_k$ avec les k premiers mots de V, suivant l'ordre établi

3. Calculer les dissimilarités inter-clusters pour chaque paire de clusters (en utilisant D'_{KL})

4. **Tant que** tous les mots de V n'ont pas été intégrés :

- Fusionner les deux clusters les plus similaires ($k-1$ clusters restant)
- Ajouter un nouveau cluster (singleton constitué du mot suivant dans V ordonné)
- Mettre à jour les dissimilarités inter-clusters.

FIG. 5.4 – Algorithme de clustering pour la méthode ADC.

$$D'_{KL}(p(C|w_t)||p(C|w_s)) = \pi_t.D_{KL}(p(C|w_t)||p(C|w_t \vee w_s)) + \pi_s.D_{KL}(p(C|w_s)||p(C|w_t \vee w_s)) \tag{5.8}$$

Dans l'équation (5.8), $p(c_j|w_t \vee w_s)$ est définie par

$$\frac{\pi_t}{\pi_t + \pi_s} p(c_j|w_t) + \frac{\pi_s}{\pi_t + \pi_s} p(c_j|w_s) \qquad \text{avec } \pi_t = p(w_t).$$

D'_{KL} vérifie les propriétés d'un indice de dissimilarité entre deux distributions. C'est cet indice que nous utiliserons dans l'approche que nous proposerons par la suite (cf. section 5.4.7).

La méthode ADC utilise donc l'indice D'_{KL} pour regrouper les mots avec l'algorithme de clustering présenté en figure 5.4.

Cet algorithme procède par fusions successives des deux plus proches clusters. Il est incrémental et donc adapté au traitement d'ensembles importants de données. La complexité de l'algorithme est en $O(lk^2m)$ avec l, k et m correspondant respectivement aux nombres de mots, clusters et classes (avec $k, m \ll l$).

5.4.6 Partitionnement par optimisation d'une fonction de qualité, issue de la théorie de l'information

La méthode ITDC (*Information Theoretic Divisive Clustering*), présentée dans [42], est basée sur l'optimisation d'un critère global d'information mutuelle, inspiré de la méthode *Information Bottleneck* [168]. Soient C la variable de classe, V le vocabulaire et $\mathcal{W}$ une partition stricte de V, l'information perdue en passant de l'espace initial issu de V, à l'espace réduit issu de $\mathcal{W}$ peut être évaluée par la différence $I(C;V) - I(C;\mathcal{W})$, où $I(X;Y)$ représente l'information mutuelle entre les deux variables aléatoires X et Y, et se définie par :

$$I(X;Y) = \sum_{x \in X, y \in Y} p(x)p(y|x) \log \frac{p(y|x)}{p(y)} \tag{5.9}$$

[42] montrent que la *fonction objective* $I(C;V) - I(C;\mathcal{W})$, peut se réécrire plus simplement par la double somme suivante :

$$I(C;V) - I(C;\mathcal{W}) = \sum_{j=1}^{k} \sum_{w_t \in W_j} \pi_t . D_{KL}(p(C|w_t)||p(C|W_j)) \tag{5.10}$$

On note au passage, que la divergence de Kullback-Leibler intervient à nouveau dans ce processus de clustering.

La figure 5.5 présente l'algorithme de clustering de la méthode ITDC. Le principe de l'algorithme s'inspire de la méthode des k-moyennes : à partir d'un ensemble de clusters, les objets sont itérativement réalloués aux centres, et ces centres sont recalculés de façon à optimiser un critère de qualité. Ici, ce critère correspond à la fonction objective définie en (5.10).

La phase d'initialisation consiste à construire des clusters "typiques" des classes cibles avant de les fusionner ou de les scinder afin d'obtenir le nombre de clusters désiré. La partition finale correspond à un optimum local pour la fonction de qualité[20]. La méthode ITDC n'est pas nécessairement moins coûteuse que la méthode ADC précédente, puisque sa complexité est en $O(lkm\tau)$ avec l, k, m et τ correspondant respectivement aux nombres de mots, clusters, classes et itérations.

5.4.7 Réduction de l'espace par pseudo-partitionnement du vocabulaire

Motivations de l'approche

Considérons un corpus d'entraînement où chaque document est étiqueté par l'une des deux classes : A ou B. Supposons que le vocabulaire V, extrait de ce corpus, s'organise en deux sous-ensembles W_1 et W_2, plus un mot w_s tels que :

- $\forall w_i \in W_1,\ p(c_A|w_i) \approx 1$ et $p(c_B|w_i) \approx 0$,
- $\forall w_i \in W_2,\ p(c_B|w_i) \approx 0$ et $p(c_B|w_i) \approx 1$,
- $p(c_A|w_s) \approx p(c_B|w_s) \approx \frac{1}{2}$.

Autrement dit, les mots de W_1 apparaissent quasi-exclusivement dans les documents de classe A, les mots de W_2 quasi-exclusivement dans les documents de classe B, tandis que le mot w_s apparaît équitablement dans les deux classes de documents A et B.

[20] Comme pour la méthode des k-moyennes, on peut montrer que cet algorithme converge vers un optimum "local", dépendant de l'initialisation.

Algorithme ITDC : *Information-Theoretic Divisive Clustering*

Entrées : Le vocabulaire V, caractérisé par :
-les distributions $\{p(C|w_t)\}_{w_t \in V}$,
-les probabilités $\{p(w_t)\}_{w_t \in V}$,
m le nombre de classes de documents,
k le nombre de clusters désiré.

Sortie : Une partition $\mathcal{W} = \{W_1, \ldots, W_k\}$ de V en k clusters disjoints.

1. Initialisation : pour chaque mot w_t, affecter w_t à W_j tel que :

$$p(c_j|w_t) = \max_{i=1,\ldots,m} p(c_i|w_t).$$

- si $k > m$, diviser arbitrairement chaque cluster en $\lfloor k/m \rfloor$ clusters (au moins),
- sinon, fusionner les clusters jusqu'à en obtenir k

2. Pour chaque cluster W_j, calculer

$$p(W_j) = \sum_{w_t \in W_j} p(w_t) \qquad \text{et} \qquad p(C|W_j) = \sum_{w_t \in W_j} \frac{p(w_t)}{p(W_j)} p(C|w_t)$$

3. Réallocation des mots : réaffecter chaque mot w_t au cluster W_{j^*} tel que

$$j^*(w_t) = \arg\min_{i=1,\ldots,k} D_{KL}(p(C|w_t)||p(C|W_i))$$

4. Si la variation de la fonction objective (5.10) est faible (e.g. $< 10^{-3}$) alors STOP
Sinon retourner à l'étape 2.

FIG. 5.5 – Algorithme de clustering pour la méthode ITDC.

Un partitionnement strict de V, conduira vraisemblablement à l'un des deux schémas suivants : $\mathcal{P}_1 = \{W_1 \cup \{w_s\}, W_2\}$ ou $\mathcal{P}_2 = \{W_1, W_2 \cup \{w_s\}\}$.

Supposons à présent que l'on cherche à classer un nouveau document d, caractérisé par la présence de w_s uniquement, parmi les mots de V.

La caractérisation d'un groupe de mot $W \subset V$ est dérivée de la caractérisation des mots qui le composent par :

$$\begin{cases} p(W_j) = \sum_{w_t \in W_j} p(w_t) \\ \\ p(c_i|W_j) = \sum_{w_t \in W_j} \frac{p(w_t)}{p(W_j)} p(c_i|w_t) \end{cases} \tag{5.11}$$

Si la partition $\mathcal{P}_1$ a été établie, on a $p(c_j|d) = p(c_j|W_1 \cup w_s)$. Par le système d'équations (5.11) et les hypothèses de distribution des mots sur les classes, on en déduit que $p(c_A|d) \gg p(c_B|d)$ d'où l'affectation de d à la classe A. En revanche, si c'est la partition $\mathcal{P}_2$ qui est établie, le document d sera classé dans B.

En généralisant cet exemple, si beaucoup de mots "ambigus" apparaîssent dans un document test, le biais induit par un partitionnement strict du vocabulaire peut entraîner une perte d'information importante dans la description des documents et ainsi produire des erreurs de classification en conséquence.

Ce phénomène peut s'expliquer également dans un cadre sémantique. Par exemple, si l'on considère que les deux classes A et B correspondent à des documents portant sur les thématiques respectives de la "justice" et de la "gastronomie". Ces deux thématiques semblent bien différentes puisqu'elles possèdent chacune leur propre terminologie. Supposons alors que les deux ensembles de mots suivants ont été extraits du corpus d'entraînement : $W_1 = \{$*Court Suprême, Magistrat, Citoyen, Avocat, ...* $\}$ et $W_2 = \{$ *Restaurant, Salade, Asperge, Avocat, ...*$\}$. Le mot $w_s =$"avocat" est polysémique et appartient aussi bien à la thématique de la "justice"[21] qu'à celle de la "gastronomie"[22]. Dans la construction d'une partition stricte du vocabulaire extrait, w_s sera inclu dans l'un des deux groupes W_1 ou W_2 exclusivement, ce qui revient à conserver l'un des deux sens observés de ce mots et à ignorer le second.

L'exemple précédent traite d'une situation extrême. Cependant on peut facilement constater que beaucoup de termes sont partagés par plusieurs documents, dans des contextes sémantiques distincts, avec différents degrés d'implication.

Nous proposons de supprimer la contrainte liée à la construction de partitions strictes, en utilisant un algorithme de clustering autorisant les recouvrements entre les clusters. Les clusters obtenus forment alors une pseudo-partition de l'ensemble des objets. Ces intersections entre clusters nécessitent une adaptation du modèle probabiliste défini jusqu'alors. Le système proposé en (5.11) se redéfini par :

$$\begin{cases} p(W_j) = \sum_{\{w_t \in V\}} p(W_j|w_t) p(w_t) \\ \\ p(c_i|W_j) = \frac{1}{p(W_j)} \sum_{\{w_t \in V\}} p(W_j|w_t) p(w_t) p(c_i|w_t). \end{cases} \tag{5.12}$$

[21] Par exemple *Le réquisitoire de l'avocat était brillant.*

[22] Par exemple *Ce restaurant propose une délicieuse salade d'avocat.*

Dans (5.12), le terme $p(W_j|w_t)$ doit être adapté au type de schéma utilisé. Jusqu'à présent, pour des schémas de partitionnement stricts, chaque objet appartient exactement à un cluster, $p(W_j|w_t)$ pouvait alors s'écrire :

$$p(W_j|w_t) = \begin{cases} 1 & si\ w_t \in W_j \\ 0 & sinon. \end{cases} \tag{5.13}$$

ce qui vérifie bien la propriété de base suivante :

$$\sum_{W_j \in \mathcal{W}} p(W_j|w_t) = 1 \tag{5.14}$$

Il reste alors à définir la probabilité $p(W_j|w_t)$ dans le cas de schémas avec recouvrements. Lorsqu'un objet appartient à plusieurs clusters, on choisira de modéliser cette probabilité par une loi uniforme. Pour un mot w_t appartenant à n clusters, on aura :

$$p(W_j|w_t) = \begin{cases} \frac{1}{n} & si\ w_t \in W \\ 0 & sinon. \end{cases} \tag{5.15}$$

Remarque. Si l'on devait définir la probabilité $p(W_j|w_t)$ dans le cas de schémas de partitionnement flous, on utiliserait les fonctions d'appartenance (probabilistes) $\{u_j(w_t)\}_{j=1...k}$ d'un mot w_t à un groupe W_j, et on aurait[23] :

$$p(W_j|w_t) = u_j(w_t) \tag{5.16}$$

$\Diamond$

Pseudo-partitionnement du vocabulaire : la méthode DDOC

Dans cette section nous présentons la méthode DDOC (*Distributional Divisive Overlapping Clustering*). DDOC reprend certains aspects des méthodes ADC et ITDC, combinés à l'algorithme de pseudo-partitionnement PoBOC. Par exemple, nous utilisons, comme dans ADC, une approche distributionnelle pour définir la similarité entre les mots du vocabulaire et une stratégie incrémentale pour regrouper ces mots. DDOC s'inspire également de la méthode ITDC, en utilisant le principe de réallocations dynamiques pour optimiser les clusters. L'algorithme de pseudo-partitionnement utilisé dans DDOC est présenté en figure 5.6.

La méthode DDOC se décompose en quatre étapes majeures :

1. Extraction d'un sous-ensemble (échantillon) V' de V,
2. Pseudo-partitionnement de V' par PoBOC,
3. Multi-affectations des mots (heuristique d'affectations de PoBOC),
4. Optimisation des clusters par réallocations dynamiques.

L'étape 1 consiste à extraire un sous-vocabulaire $V' \subset V$. Pour cela, les mots de V sont ordonnés par information mutuelle $I(w;C)$ décroissante avec la variable de classe. Cette technique d'ordonnancement, également utilisée dans la phase d'initialisation de la méthode ADC, permet de sélectionner ensuite les mots les plus influents dans la description des classes de documents. Les M premiers mots sont retenus pour constituer V' (M spécifié par l'utilisateur).

[23] Sous réserve qu'il s'agisse bien de fonctions d'appartenance probabilistes ($\sum_j u_j(w_t) = 1$).

Algorithme DDOC : *Distributional Divisive Overlapping Clustering*

Entrées : Le vocabulaire V, caractérisé par :
-les distributions $\{p(C|w_t)\}_{w_t \in V}$,
-les probabilités $\{p(w_t)\}_{w_t \in V}$,
l le nombre de classes de documents,
M, f et τ trois paramètres fixés
(M : taille du sous-vocabulaire, f : fuzzifier, τ : nb. itérations).

Sortie : Une pseudo-partition $\mathcal{W}$ de V en clusters non-disjoints.

Étape 1.
Ordonner V *via* l'information mutuelle avec la variable de classe $I(w; C)$
Construire le sous-vocabulaire $V' \subset V$, constitué des M premiers mots de V (selon l'ordre établi)
Construire la matrice de dissimilarité D sur V' (par 5.8) telle que

$$d(w_t, w_s) = D'_{KL}(p(C|w_t)||p(C|w_s))$$

Étape 2.
Exécuter `PoBOC`(V', D) afin d'obtenir une pseudo-partition $\mathcal{W}'$ de V'
Pour chaque cluster $W_j \in \mathcal{W}$, calculer $p(W_j)$ et $p(C|W_j)$ par 5.12

Étape 3.
Pour chaque mot $w_t \in V \setminus V'$ (dans l'ordre établi) :
- `Affecter`$(w_t, \mathcal{W}, f)$ (cf. figure 5.7)
- Pour chaque cluster W_j modifié, recalculer $p(W_j)$ et $p(C|W_j)$

Étape 4.
Tant que $\mathcal{W}$ est modifiée et moins de τ itérations :
Pour chaque mot $w_t \in V$: `Affecter`$(w_t, \mathcal{W}, f)$
Pour chaque cluster $W_j \in \mathcal{W}$: recalculer $p(W_j)$ et $p(C|W_j)$

FIG. 5.6 – Algorithme de clustering pour la méthode DDOC.

Procedure `Affecter`$(w_t, \mathcal{W}, f)$

Soient la pseudo-partition $\mathcal{W} = \{W_1, \ldots, W_k\}$, et w_t un mot à affecter

1. Pour chaque cluster W_j, calculer la similarité $d(w_t, W_j)$ par

$$s(w_t, W_j) = 1 - D'_{KL}(p(C|w_t)||p(C|W_j))$$

2. Ordonner les clusters en construisant la fonction bijective d'ordonnancement $\Phi : \{1, \ldots, k\} \rightarrow \{1, \ldots, k\}$ telle que :

$$s(w_t, W_{\Phi(1)}) \geq s(w_t, W_{\Phi(2)}) \geq \cdots \geq s(w_t, W_{\Phi(k)})$$

3. $j = 1$, affecter w_t à $W_{\Phi(j)}$ $(W_{\Phi(j)} = W_{\Phi(j)} \cup w_t)$

4. **Tant que** w_t a été affecté à $W_{\Phi(j)}$ et $j < k$:
   $j = j + 1$
   Si $s(w_t, W_{\Phi(j)}) > (1 - f).s(w_t, W_{\Phi(j-1)}) + f.s(w_t, W_{\Phi(j+1)})$ affecter w_t à $W_{\Phi(j)}$
   Si $j = k$ et $s(w_t, W_{\Phi(j)}) > (1 - f).s(w_t, W_{\Phi(j-1)})$ affecter w_t à $W_{\Phi(j)}$

FIG. 5.7 – Procédure de multi-affectations utilisée dans la méthode DDOC.

L'étape 2 exécute l'algorithme PoBOC sur l'ensemble V', auquel est associé une matrice de dissimilarité D de taille $M \times M$ et telle que $d(w_s, w_t) = D'_{KL}(p(C|w_s)||p(C|w_t))$. V' est alors partitionné en k clusters non-disjoints, avec k déterminé automatiquement. Cette étape peut être vue comme l'initialisation de l'algorithme et cette initialisation diffère de l'approche ADC, qui débute avec uniquement des singletons, et de l'algorithme ITDC qui scinde ou fusionne "arbitrairement" une première partition de V afin d'obtenir le nombre désiré de clusters. Parceque les groupes constitués sont non-disjoints et que le système propose automatiquement un nombre approprié de groupes, on peut supposer que cette initialisation est à la fois plus précise et plus proche de l'organisation réelle des mots entre eux.

L'étape 3 utilise l'heuristique de multi-affectations de PoBOC (cf. chapitre 2) pour intégrer les $|V| - M$ mots restants (en suivant l'ordre établi) parmi les k clusters initiaux. Nous choisissons de contrôler les multi-affectations par l'utilisation d'un paramètre (f) dont le principe s'apparente à celui d'un *fuzzifier* pour les méthodes de clustering flou. Pour $f = 0$ la partition obtenue est stricte, le nombre d'affectations augmente lorsque f augmente, jusqu'à $f = 1$ et dans ce cas chaque objet est affecté à chaque clusters[24]. Notons que pour $f = 1/2$, on se ramène aux conditions d'affectation par défaut, comme présenté dans le chapitre 2. Nous redéfinissons la procédure d'affectations dans la figure 5.7, en intégrant ce paramètre.

Notons que la stratégie globale d'affectations est adaptative, autrement-dit, les caractéristiques des clusters sont systématiquement recalculées après l'ajout d'un objet. On sait que les approches adaptatives permettent d'accéder plus rapidement à un

[24]Tous les clusers sont identiques et contiennent tous les objets.

bon schéma de clustering (cf. version adaptative de k-moyennes, chapitre 1).

L'étape 4 "optimise" les clusters générés par réallocations (multiples) dynamiques des objets aux clusters. Il s'agit en fait de rectifier les groupes obtenus par le processus incrémental de l'étape précédente. En effet, l'ajout répétitif d'objets dans un clusters, modifie progressivement les caractéristiques de celui-ci, si bien qu'un objet constituant le cluster initial ou faisant parti des premiers objets ajoutés, peut, à la fin du processus incrémental, ne plus y trouver sa place (dissimilarité élevée entre la distribution du mot et celle du cluster). Remarquons que l'ordonnancement des mots, établi à l'étape 1, peut aider à limiter le phénomène que nous venons d'identifier.

En considérant que la taille du sous-vocabulaire V' est négligeable par rapport à la taille du vocabulaire initial V, la complexité de la méthode DDOC est en $O(lkm\tau)$ avec l, k, m et τ correspondant respectivement aux nombres de mots, clusters, classes et itérations.

5.4.8 Étude expérimentale

Objectifs de l'étude

Deux objectifs sont visés dans cette étude, le premier concerne une avancée dans le domaine de la Recherche d'Information, le second porte sur l'ensemble du sujet abordé dans cette thèse.

D'un point de vue RI, on cherche à valider l'hypothèse selon laquelle, la construction de groupes non-disjoints de mots plutôt que de groupes disjoints, permettrait de mieux décrire les documents, dans une perspective de classification supervisée. Pour cela, nous observerons, sur une même méthode de réduction (ici DDOC), l'influence des intersections sur le résultat final de la classification. Sur la méthode DDOC, nous ferons varier le fuzzifier f pour augmenter ou diminuer l'importance des multi-affectations.

Plus généralement, cette étude dans le domaine de la RI est le contexte idéal pour l'évaluation de l'algorithme de clustering PoBOC, présenté dans le chapitre 2. En effet, les relations sémantiques complexes qu'entretiennent entre eux les objets textuels, font de ce type de données une cible appropriée pour montrer l'intérêt des méthodes de clustering produisant des clusters non-disjoints. L'algorithme PoBOC sera alors comparé, *via* la méthode DDOC, à deux algorithmes traditionnels de regroupement en classes disjointes : une approche hiérarchique et une approche par partitionnement à travers les méthodes ADC et ITDC respectivement.

Notons que pour ces expérimentations, nous évaluerons de façon indirecte la qualité des algorithmes de clustering, en observant la performance du classifieur naïf de Bayes sur différents schémas de représentation des documents.

Présentation des corpus

Nous utilisons deux corpus classiques, faisant référence dans le domaine de la classification de documents : le corpus *20Newsgroup* [104] et le corpus *Reuters-21578*[25].

20Newsgroup est un corpus constitué de 18941 articles. Il s'agit d'échanges entre personnes dans le cadre d'un forum de discussions[26]. Les documents sont organisés autour de 20 thématiques (groupes de discussions) qui constituent alors les 20 classes à apprendre. Deux sous-corpus sont prédéfinis : l'ensemble des documents d'entraînement (60%) et de

[25] www.research.att.com/∼lewis/reuters21578.html

[26] Nous utilisons la version "by date" de ce corpus (www.ai.mit.edu/∼jrennie/20Newsgroups/)

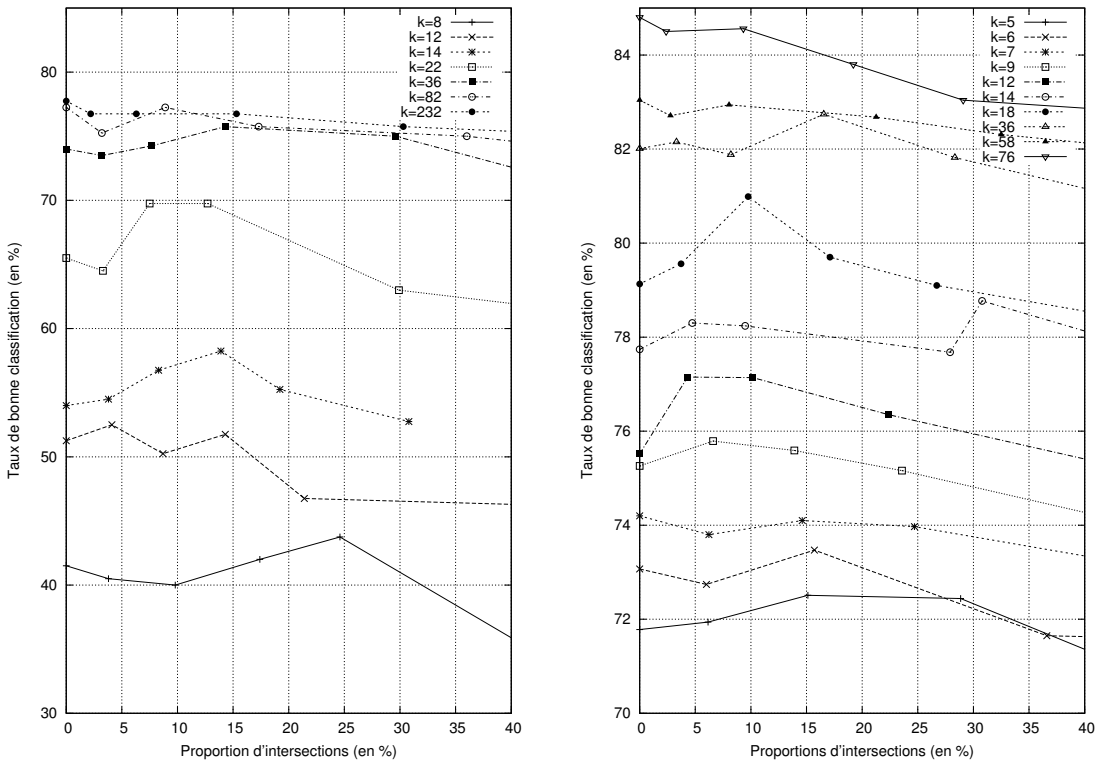


FIG. 5.8 – Évaluation de l'influence de la proportion d'intersections entre clusters, sur la performance du classifieur. Corpus *Newsgroup* (gauche) et *Reuters* (droite).

test (40%). A partir du corpus d'entraînement, 22183 mots sont extraits par stemmatisation et suppression des mots vides (utilisation d'une *stoplist*) et des mots apparaissant dans moins de trois documents.

Le corpus *Reuters-21578* se divise en 9603 documents pour l'ensemble d'entraînement et 3299 documents tests, en utilisant la collection modifiée "ModApte" [6]. Ces documents sont organisés autour de 118 classes non-exclusives (un document peut être étiqueté par plusieurs classes différentes). Par un processus similaire au précédent, 7680 mots sont extraits du corpus d'entraînement.

Calcul de la performance des classifieurs

Nous nous plaçons dans le cas d'une classification binaire simple, c'est à dire que, pour chaque document test d à classer, nous retenons la classe d'appartenance la plus probable c^*, calculée par le classifieur naïf de Bayes :

$$c^*(d) = \arg\max_{i=1...m} p(c_i|d)$$

Si cette classe correspond à l'étiquette (ou l'une des étiquettes) proposée(s) pour ce document, on le considère bien classé. La performance du classifieur est alors quantifiée par le rapport du nombre de documents bien classés sur le nombre total de documents tests.

L'évaluation est effectuée sur une exécution, en utilisant le découpage pré-défini des ensembles d'entraînement et de test. Précisons que les paramètres du classifieur de Bayes

sont appris sur les corpus d'entraînement uniquement, puis évalués sur les corpus de test. Il en va de même pour l'étape de clustering supervisée basée sur des mesures liées aux distributions des mots sur les classes (divergence de KL, fonction objective de ITDC, etc.).

Influence des intersections entre groupes de mots

Sur les deux corpus on observe l'influence des intersections entre clusters sur les performances du classifieur induit (figure 5.8). Les "proportions d'intersections" sont calculées sur le pseudo-partitionnement final $\mathcal{W}$ du vocabulaire initial V :

$$Int(\mathcal{W}) = \frac{\sum_{W_j \in \mathcal{W}} |W_j|}{|V|} - 1$$

Pour $f = 0$, le schéma de partitionnement obtenu est strict donc $\sum_{W_j \in \mathcal{W}} |W_j| = |V|$ et la proportion d'intersections est nulle. Lorsque f augmente, cet indice augmente également.

On observe sur la figure 5.8 que la tendance générale des tracés, est identique sur les deux corpus, à savoir une amélioration du classifieur lorsqu'on autorise jusqu'à 10% à 20% d'intersections, puis une dégradation de ses performances, au delà de 30%. On note également que les améliorations indiquées sont plus importantes pour des schémas constitués de peu de clusters (jusqu'à 20 clusters) et disparaissent progressivement lorsque le nombre de clusters augmente.

On déduit de cette première étude, que le pseudo-partitionnement du vocabulaire est particulièrement indiqué lorsque l'on souhaite décrire l'ensemble des documents avec peu d'attributs. D'autre part, il existe certaines conditions à vérifier pour que les intersections entre clusters influencent positivement la description des documents, et donc indirectement la performance du classifieur. Ces conditions portent notamment sur la proportion d'intersections autorisées. Empiriquement, les expériences réalisées sur les deux corpus s'accordent sur une proportion comprise entre 10% et 20%. Cet intervale de valeurs est globalement respecté pour le paramétrage $f = 0.3$ sur le corpus *Newsgroup* et $f = 0.2$ sur le corpus *Reuters*. Nous retenons donc ces deux paramètres, pour comparer DDOC avec ADC et ITDC, dans l'expérience suivante.

Comparaison des méthodes ADC, ITDC et DDOC

La seconde expérience permet de comparer la méthode DDOC avec les deux autres approches de réduction présentées : ADC (clustering hiérarchique) et ITDC (clustering par partitionnement). Cette comparaison est effectuée dans des configurations identiques (nombre de clusters, corpus d'entraînement, de test, vocabulaire et classifieur identiques). La différence concerne alors l'approche utilisée pour parvenir à un partitionnement approprié du vocabulaire.

Les résultats de cette étude sont présentés dans les diagrammes de la figure 5.9. Sur le corpus *Newsgroup* (diagramme de gauche), on observe une performance sensiblement meilleure pour la méthode que nous proposons, particulièrement pour un nombre de clusters limité (jusqu'à 36 clusters). C'est en partie sur ce corpus que la méthode ITDC est évaluée dans [42]. On observe effectivement, que le regroupement agglomératif proposé par ADC est légèrement moins pertinent que le partitionnement obtenu par ITDC.

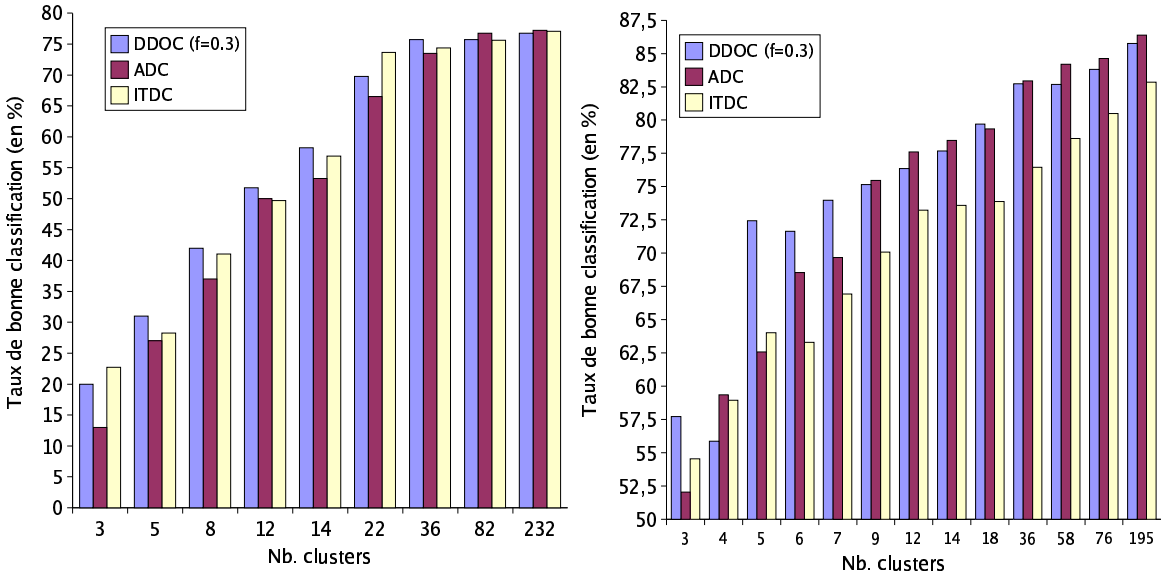


FIG. 5.9 – Évaluation de la méthode DDOC, comparativement aux méthodes ADC et ITDC sur les deux corpus : *Newsgroup* (gauche) et *Reuters* (droite).

En revanche, sur le corpus *Reuters* (diagramme de droite), la méthode ITDC est nettement moins performante que les deux autres approches[27]. Sur ce même corpus, le pseudo-partitionnement proposé par l'algorithme PoBOC est réellement performant, toujours pour des valeurs de k (nombre de groupes) limitées.

Par l'étude décrite dans cette section (5.4), nous avons proposé une avancée dans le domaine de la recherche d'information, en présentant la méthode DDOC permettant de réduire l'espace de description des documents par un pseudo-partitionnement du vocabulaire. Les expérimentations effectuées donnent des résultats encourageants, mais devront être complétées, en utilisant par exemple, d'autres systèmes de classification (SVM, arbres de décision, etc.) ainsi que des mesures d'évaluation plus précises (e.g. micro/macro précision et rappel). Une étude devra également porter sur la recherche voire l'ajustement automatique du paramètre f.

Plus généralement, ce travail nous a permi de confirmer l'hypothèse supportant ce chapitre, à savoir qu'il est pertinent d'organiser les données textuelles en classes non-disjointes. Par la même occasion, l'algorithme PoBOC a pu, une nouvelle fois être évalué avec succès.

5.5 Étude prospective : application à l'analyse des genres

Nous présentons dans cette dernière section, les premiers résultats d'un étude effectuée en collaboration avec une équipe de linguistes. L'objectif de cette étude est d'évaluer

[27] Notons que la comparaison entre les deux approches ADC et ITDC sur le corpus *Reuters* n'a jamais été présentée jusqu'à ce jour.

l'influence du genre dans le processus de classification thématique de documents (on parle également de classification par domaines). Ce travail est avant tout motivé par l'idée que chaque thématique est caractérisée par un type de discours[28] qui lui est propre. De façon plus schématique, nous pensons qu'il existe certains descripteurs morphologiques et/ou syntaxiques permettant de distinguer, au moins partiellement, un discours portant sur un thème A, d'un discours portant sur un thème B. Il s'agit là d'une hypothèse lourde de conséquences selon deux points de vue.

D'une part du point de vue de la linguistique, puisque l'hypothèse d'une indépendance genre/domaine est généralement admise, d'autre part en recherche d'information où l'apparition de nouveaux descripteurs pourrait permettre d'améliorer encore les performances des classifieurs thématiques. Les conditions d'intégration de ces nouvelles variables resteront alors à définir.

Pour évaluer l'hypothèse avancée, nous avons proposé une analyse exploratoire d'un corpus d'articles scientifiques issus du domaine de la linguistique. Du fait de la thématique de ce corpus, nous disposons d'experts capables d'évaluer la cohérence d'un résultat (par exemple d'une sous-classe d'articles). Les documents, après analyse morphosyntaxique, ont été représentés sur un ensemble de 130 variables descriptives (pourcentages de noms, verbes, ponctuations, etc.). Deux sous-ensembles de variables ont été extraits de cet ensemble initial, à partir d'une réflexion de la part des experts, appuyée par une étude statistique fine (analyse des corrélations, analyse en composantes principales, etc.).

A partir de ces différentes descriptions, nous avons utilisé une distance classique (dans un premier temps la distance euclidienne), pour proposer trois schémas de classification *via* PoBOC. Notre approche de regroupement intéresse effectivement les experts pour trois raisons :

- **Le nombre de groupes** étant proposé automatiquement, cela évite une analyse fastidieuse qui consiste généralement à envisager différents schémas pour un nombre de clusters évoluant de 5 en 5, en laissant l'expert rechercher une organisation qui puisse convenir ;
- **Les intersections** entre les groupes sont adaptés à la tâche de regroupement de documents, ces derniers peuvent en effet contenir plusieurs thématiques (ou sous-thématiques) ou encore se situer entre deux genres ;
- **Le traitement des outliers** effectué par PoBOC, aide l'expert à identifier des documents atypiques afin de les étudier en particulier ou, au contraire de les supprimer de son corpus afin de l'homogénéiser.

Nous donnons en annexe D une partie de l'analyse proposée par les experts, sur les classifications obtenues à partir d'un ensemble de 224 documents. Globalement "les classes obtenues semblent reliées thématiquement" et 14 clusters sur 44 (non réduits à un seul documents) sont même jugés "thématiquement pertinents" (documents issus d'une même revue voire d'un même numéro thématique). Quelques descripteurs semblent jouer un rôle plus important dans la constitution de ces classes, comme par exemple les ponctuations, dates, noms propres et verbes (notamment leur temps).

Un nombre non négligeable de documents donnent lieu à des clusters "singletons" (entre 6 et 14 selon la description utilisée). De par notre connaissance de l'algorithme PoBOC, on peut considérer que ces documents sont des "outliers", autrement dit des documents atypiques relativement au reste du corpus. En observant que la plupart de ces

[28] On entend par discours, une production en langue naturelle, au sens large.

documents proviennent des mêmes revues, l'expert pourrait être amené à reconsidérer la légitimité de cette source dans la constitution de son corpus.

Cette première série d'expérimentations laisse penser à une validation, au moins partielle, de l'hypothèse avancée, sur l'intégration du genre dans la définition du domaine. La poursuite de cette étude concernera l'analyse (voire la confrontation avec) d'autres corpus de documents moins spécialisés, l'intégration d'autres outils émanant du domaine de l'apprentissage comme par exemple la sélection ou pondération automatique de variables pertinentes, ou encore l'apprentissage de règles de caractérisation afin d'assister l'expert dans son analyse.

Conclusion

Contributions

Nous nous sommes intéressés, dans cette thèse, à la construction de schémas de classification constitués de groupes non-disjoints. Cette étude est motivée par le constat que les méthodes d'apprentissage non-supervisé génèrent des schémas stricts ou flous, sans proposer d'alternatives, autres que dans le cas d'applications spécifiques. La construction de schémas avec recouvrements est indispensable dans certaines situations, et souhaitable dans beaucoup d'autres.

Notre première contribution concerne la proposition de l'algorithme PoBOC (*Pole-Based Overlapping Clustering*) [30, 31]. Cet algorithme de clustering est basé sur la recherche de pôles, définis par des cliques dans un graphe de dissimilarité, auxquels sont attribués les objets lors d'une phase de "multi-affectation". Outre le fait qu'il produise un pseudo-partitionnement de l'ensemble des objets, cet algorithme présente certaines caractéristiques intéressantes telles que : la prise en compte des voisinages locaux pour générer des groupes de densités variées ou le choix automatique d'un nombre approprié de groupes, évitant ainsi le paramétrage contraignant des algorithmes du type k-moyennes.

Nous avons tenté de justifier intuitivement et théoriquement les différentes étapes du processus de regroupement effectué par PoBOC, avant d'évaluer les résultats obtenus sur des jeux de données classiques en apprentissage (Iris, Zoology, Soybean, etc.). Ces résultats sont apparus satisfaisants, comparativement à l'algorithme de référence des k-moyennes, alors même que les seules mesures de qualité existantes et sur lesquelles nous nous sommes basés, ont tendance à favoriser les schémas de clusters disjoints.

Dans un deuxième temps, nous avons présenté un système d'apprentissage de règles par décomposition des classes ou concepts. D'abord dans un formalisme propositionnel [30, 31], puis étendu aux règles formalisées en logique du premier ordre [29], le système RuLe-Clust (*Rule-Learning from Clustering*) consiste à décomposer chaque classe ou concept cible, en sous-ensembles par un processus de clustering. Chacun de ces sous-ensembles guide, ensuite, l'étape de construction d'une règle.

Dans le cas attribut/valeur, nous avons proposé une série d'expérimentations permettant de conclure à l'intérêt d'organiser une classe d'exemples avant de chercher à la couvrir par une ou plusieurs règles. Par ailleurs, les résultats obtenus en utilisant l'algorithme PoBOC pour le processus de décomposition sont sensiblement meilleurs que pour d'autres approches traditionnelles de regroupement. Nous avons tenté de cerner les raisons pour

lesquelles PoBOC est davantage approprié et avons observé que ce résultat est, en partie, dû au type de schéma proposé (groupes non-disjoints).

L'extension à l'apprentissage de règles logiques est basé sur le même principe. La structuration des exemples positifs du concept cible guide la recherche des littéraux discriminants pour aboutir à une hypothèse disjonctive pertinente, mettant en exergue des sous-concepts naturels, relativement au langage de représentation choisi. Nous avons montré que l'algorithme PoBOC offre une décomposition satisfaisante puisqu'elle permet généralement de retrouver les sous-concepts (éventuellement non-disjoints) attendus.

Enfin, nous nous sommes concentrés sur une application particulière en considérant le problème de la construction de classes contextuelles de mots, dans le cadre de la recherche d'information [25, 27]. Sur un ensemble de données de taille réduite, nous avons observé et comparé des schémas avec et sans recouvrements entre les groupes. Dans cette application, les schémas autorisant les intersections entre les classes contextuelles sont justifiés car ils permettent, contrairement aux schémas stricts, de préserver les différents sens des mots. Nous avons été amené, pour cela, à définir un nouvel indice de proximité, dont l'originalité est d'utiliser des connaissances extraites du Web pour évaluer la proximité sémantique/contextuelle de deux mots.

Encouragés par ces premiers résultats, nous avons proposé la méthode DDOC (*Distributional Divisive Overlapping Clustering*) [26, 28], pour réduire l'espace de description des documents par regroupement des mots en groupes non-disjoints. Cette méthode s'appuie sur une version incrémentale de l'algorithme PoBOC, et permet d'extraire des attributs pertinents pour décrire les documents, dans une perspective de classification supervisée. L'évaluation proposée sur les corpus *Reuters-21578* et *20Newsgroup* a montré que, sous certaines conditions (réduction importante de l'espace et intersections contrôlées), les nouveaux attributs extraits par la méthode DDOC améliorent la justesse du classifieur, comparativement aux approches agglomératives et divisives faisant référence dans ce domaine. Une nouvelle fois, l'organisation des mots en groupes non-disjoints est l'une des explications de ce phénomène.

Perspectives

Les perspectives pour ce travail sont très nombreuses. Sur chaque méthode ou système proposé, des adaptations ou évaluations complémentaires sont envisagées.

Tout d'abord, l'algorithme PoBOC, utilisé comme outil de validation de l'hypothèse de départ, pourrait faire l'objet d'adaptations. Nous étudierons de façon plus formelle, la variante incrémentale introduite dans le contexte de la méthode DDOC, dans le but de la généraliser au traitement de tous types de données. Un autre aspect à analyser concerne la notion de voisinage, utilisée dans la définition du graphe de dissimilarité, qui pourrait être réduite à un sous-voisinage, de manière à limiter l'influence des objets très éloignés sur le voisinage d'un objet. Enfin, le critère permettant de contrôler l'importance des intersections, introduit dans l'approche DDOC, sera étudié et probablement intégré à l'algorithme PoBOC, dans une perspective expérimentale.

Le système RuLe-Clust nous apparaît également comme une approche prometteuse. Une première évolution du système, mentionnée dans le chapitre 3, consistera à limiter le phénomène de sur-adéquation aux données d'entraînement, par le biais de deux adaptations probablement complémentaires : d'une part en acceptant une certaine tolérance

dans la couverture des exemples d'un cluster[29] et d'autre part en proposant une étape d'élagage *a posteriori*, sur l'ensemble des règles apprises.

Une autre étude portera sur le test d'existence d'une règle couvrant l'ensemble des exemples d'un cluster et rejetant tous les contre exemples du concept cible. En effet, ce test permet à la fois de valider la décomposition et de gagner en complexité en évitant de lancer la recherche d'une règle si elle n'existe pas. Malheureusement, si l'existence d'une telle règle peut être facilement testée dans le formalisme propositionnel, en logique du premier ordre, l'espace de recherche peut être infini. Nous envisageons alors une stratégie basée sur le test de couverture des exemples négatifs par un plus petit généralisé d'exemples d'un groupe.

En recherche d'information, nous compléterons, dans un premier temps, les évaluations effectuées pour la méthode DDOC, en couplant ce processus avec un classifieur SVM plutôt que le classifieur naïf de Bayes. Les performances de l'approche SVM restent inégalées sur cette application [93] et nous espérons pouvoir accentuer encore le gain induit par notre méthode d'extraction d'attributs.

Parallèlement, nous analyserons la méthode DDOC d'un point de vue formel, en théorie de l'information. Par exemple, nous observerons l'influence des intersections entre les groupes de mots à l'aide de mesures de gain d'information ou d'information mutuelle.

Nous poursuivrons également, l'étude prospective engagée, destinée à démontrer empiriquement l'intérêt d'utiliser des descripteurs du genre pour améliorer la classification thématique de documents [145]. Ce travail, effectué en étroite collaboration avec des spécialistes de la statistique lexicale et de la linguistique, cherchera de plus à dégager un ensemble de descripteurs morphologiques ou syntaxiques pertinents pour cette tâche.

Enfin, nous espérons par cette étude, avoir accompagné l'évolution actuelle qui consiste à ne plus se contenter de structurations simples en groupes homogènes et bien séparés mais plutôt à considérer d'autres schémas de classification, plus souples, plus riches et plus représentatifs des relations partagées par les objets. Nous souhaitons alors que l'intérêt croissant pour ce type d'approche se manifeste par la proposition d'autres algorithmes, plus rapides notamment. Ce développement est conditionné par l'existence d'un outil permettant de comparer les méthodes entre elles. Nous envisageons alors de proposer une mesure de qualité adaptée à l'évaluation de pseudo-partitions.

[29] Autoriser une faible proportion d'exemples positifs (resp. négatifs) non-couverts (resp. couverts).

A

Mesures d'évaluation des schémas de clustering

Table des symboles :

n : le nombre d'objets à traiter,
$\mathcal{C}$: le schéma de clustering à évaluer,
$\mathcal{P}$: la classification de référence (connaissance extérieure),
M : le nombre total de paires d'objets ($M = n.(n-1)/2$),
D : une matrice de dissimilarité sur les données,
P_C : la matrice cophénétique de $\mathcal{C}$ (dans le cas d'une hiérarchie),
$X(i,j)$: distance entre les clusters contenant x_i et x_j dans $\mathcal{C}$,
$Y(i,j)$: distance entre les clusters contenant x_i et x_j dans $\mathcal{P}$,
a : nombre de paires d'objets dans un même cluster dans $\mathcal{C}$ et dans $\mathcal{P}$,
b : nombre de paires d'objets dans un même cluster dans $\mathcal{C}$ et non dans $\mathcal{P}$,
c : nombre de paires d'objets dans un même cluster dans $\mathcal{P}$ et non dans $\mathcal{C}$,
d : nombre de paires d'objets dans des clusters différents dans $\mathcal{C}$ et dans $\mathcal{P}$,
$k_{\mathcal{C}}$: nombre de clusters dans $\mathcal{C}$,
$k_{\mathcal{P}}$: nombre de clusters dans $\mathcal{P}$,
p_j : rapport entre le nombre d'objets de classe j dans $\mathcal{P}$, présents dans C_i et la taille de C_i,
$\delta_{i,j}$: égal à 1 si x_i et x_j appartiennent au même cluster dans $\mathcal{C}$,
$V(C_i)$: variance du cluster C_i,
x_i^* : représentant (centroïde ou médoïde) du cluster C_i,
$u_k(x_i)$: valeur d'appartenance de x_i au cluster C_k,

Indices d'évaluation externe

Statistique de *Huberts*	$\Gamma(\mathcal{C},\mathcal{P}) = \frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X(i,j).Y(i,j)$
Statistique de *Rand*	$R(\mathcal{C},\mathcal{P}) = \frac{a+d}{a+b+c+d}$
Statistique de *Jaccard*	$J(\mathcal{C},\mathcal{P}) = \frac{a}{a+b+c}$
Indice de *Fowlkes et Mallows*	$FM(\mathcal{C},\mathcal{P}) = \sqrt{\frac{a}{a+b}\ \frac{a}{a+c}}$
Mesure d'entropie	$E(\mathcal{C},\mathcal{P}) = \sum_{i=1}^{k_\mathcal{C}} \frac{\lvert C_i\rvert.\varphi(C_i)}{n}\ \ ;\ \ \varphi(C_i) = -\sum_{j=1}^{k_\mathcal{P}} p_j.\log p_j$

Indices d'évaluation interne

Statistique de *Huberts*	$\Gamma(\mathcal{C},D) = \frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d(x_i,x_j).(1-\delta_{i,j})$
Coefficient de corrélation cophénétique	$CPCC(P_C,D) = \frac{\frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P(i,j).P_C(i,j) - \mu_P\mu_C}{\sqrt{\left[\frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P(i,j)^2 - \mu_P^2\right]\left[\frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P_C(i,j)^2 - \mu_C^2\right]}}$

Indices d'évaluation relative

Statistique de *Huberts*	$\Gamma(\mathcal{C}) = \frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d(x_i,x_j).X(i,j)$
Indice de *Dunn*	$D(\mathcal{C}) = \min_{i=1\ldots t}\min_{j\neq i}\left(\frac{\min_{x\in C_i, y\in C_j} d(x,y)}{\max_{k=1\ldots t} diam(C_i)}\right)$
Indice de *Davies et Bouldin*	$DB(\mathcal{C}) = \frac{1}{t}\sum_{k=1}^{t}\max_{j\neq k}\frac{V(C_k)+V(C_j)}{d(x_k^*,x_j^*)}$

Indices d'évaluation pour schémas flous

Indice de séparation	$S(\mathcal{C}) = \dfrac{\sum_{k=1}^{t}\sum_{i=1}^{n} u_k^2(x_i) d(x_i, x_k^*)^2}{n. \min_{i \neq j} d(x_i^*, x_j^*)^2}$
Coefficient de partition	$PC(\mathcal{C}) = \frac{1}{n} \sum_{i=1}^{n}\sum_{k=1}^{t} u_k^2(x_i)$

B

Évaluation du nombre de clusters

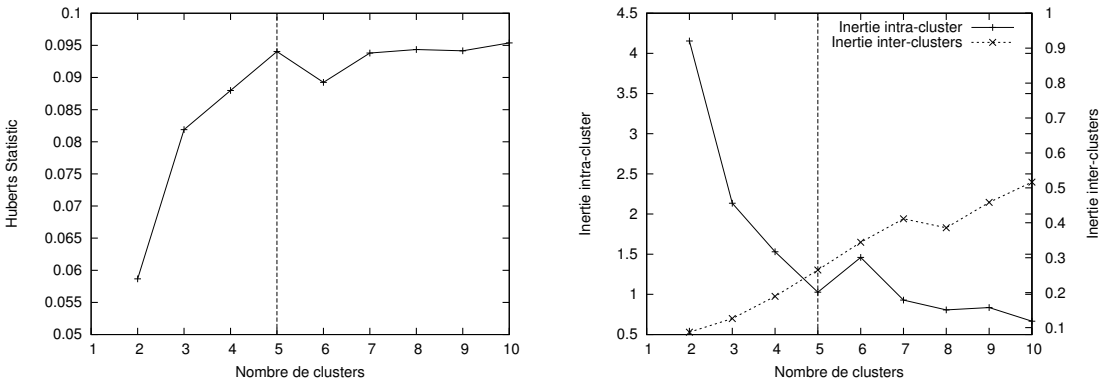


FIG. B.1 – Zoology.

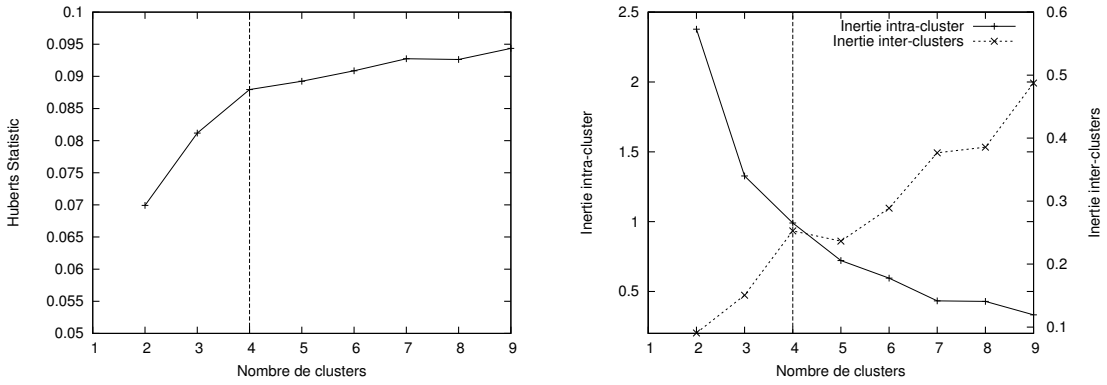


FIG. B.2 – Wine.

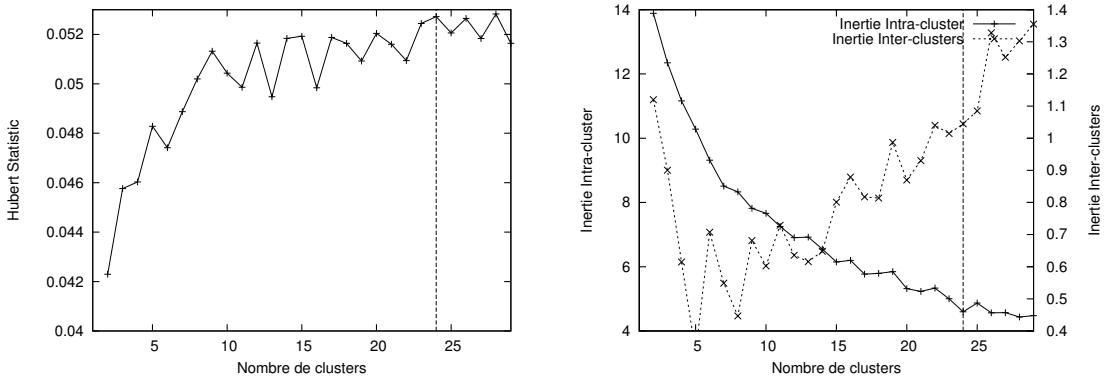


FIG. B.3 – Soybean.

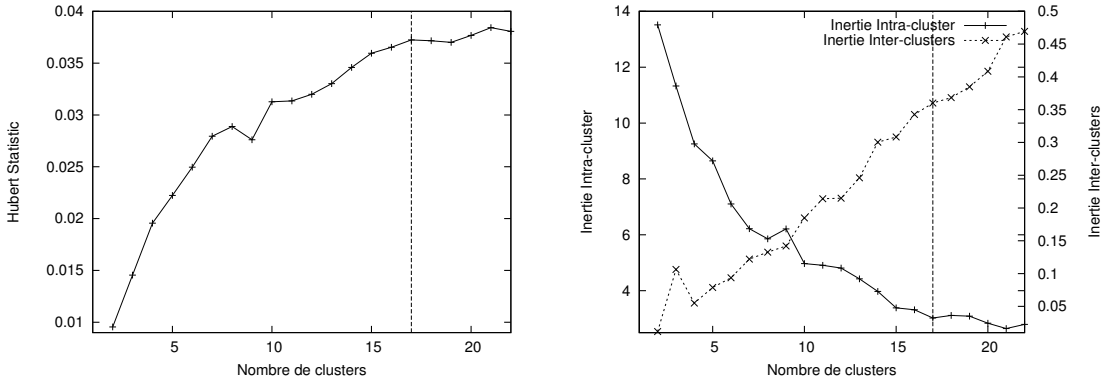


FIG. B.4 – Annealing.

C

Mesures de qualité d'une règle de classification

Cette annexe vise à définir les différentes mesures de performance, traditionnellement utilisées dans les systèmes de classification à partir de règles.

Soit $E = \{e_1, \ldots, e_N\}$ un ensemble d'entraînement, contenant N exemples, étiquetés chacun par l'une des classes $c_1, \ldots, c_k$.

Typiquement, une règle R est performante si elle couvre beaucoup d'exemples d'une même classe et peu d'exemples des autres classes. Une manière d'évaluer et de comparer la performance des règles est alors de calculer la *fréquence relative*.

• Soit n_1 le nombre d'exemples couverts par la règle R et n_1^1 la quantité d'exemples classés correctement (nombre d'exemples appartenant à la classe donnée par R), la **fréquence relative** estime la performance de la règle R par

$$f(R) = \frac{n_1^1}{n_1}$$

La fréquence relative est utilisée pour évaluer les règles générées par le système AQ [123], par exemple.

En considérant la notation $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ où C_i correspond à l'ensemble des exemples étiquetés c_i (définition en extension), $\mathcal{C}$ peut être vue comme la partition de référence de E. Une autre manière d'évaluer la performance d'une règle R, est de mesurer l'*entropie* de la couverture de R, relativement à la partition de référence $\mathcal{C}$.

• Soit $cov(R)$ la couverture d'une règle R, on note p_i la proportion d'exemple de classe c_i dans $cov(R)$ ($p_i = \frac{|cov(R) \cap C_i|}{|cov(R)|}$). L'**entropie** de $cov(R)$ mesure la pureté de cet ensemble d'exemples, relativement aux classes attendues :

$$Entropie(cov(R)) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

Une valeur nulle, pour cette mesure, indique que tous les exemples couverts par R appartiennent à la même classe. A l'inverse, une valeur élevée (égale à 1 dans le cas de 2 classes) indique que R couvre uniformément chaque classe et que la règle est peu pertinente.

L'entropie est un critère utilisé dans d'autres mesures de performance que nous définissons dans la suite. Ces mesures sont basées sur une classification binaire. Dans ce cas, les exemples positifs E^+ sont ceux de la classe ciblée par la règle apprise R, et l'ensemble des exemples négatifs correspond au complémentaire $E^- = E \setminus E^+$ (autrement dit, les exemples de toutes les autres classes).

Pour une règle R, on peut dresser le tableau de contingence (tableau C.1) visant à observer la répartition des exemples sur les deux paires d'ensembles (E^+, E^-) et $(cov(R), \overline{cov(R)})$.

	E^+	E^-	
$cov(R)$	n_1^1	n_1^0	n_1
$\overline{cov(R)}$	n_0^1	n_0^0	n_0
	n^1	n^0	N

TAB. C.1 – Table de contingence et notations.

Dans ce tableau, n_1^1 désigne le nombre d'exemples positifs couverts par la règle R, n_1^0 le nombre d'exemples négatifs couverts par R, n_0^1 le nombre d'exemples positifs non-couverts par R et n_0^0 le nombre d'exemples négatifs non-couverts par R. Enfin, n^1 et n^0 désignent respectivement le nombre total d'exemples positifs et négatifs, tandis que n_1 et n_0 correspondent au nombre d'exemples couverts ou non-couverts par R.

On considère également les notations suivantes :

$P^1 = \frac{n^1}{N}$; $P^0 = \frac{n^0}{N}$: Taux d'exemples positifs et négatifs sur E,
$P_1 = \frac{n_1}{N}$; $P_0 = \frac{n_0}{N}$: Taux d'exemples couverts et non-couverts par R,
$P_1^1 = \frac{n_1^1}{n_1}$; $P_1^0 = \frac{n_1^0}{n_1}$: Taux d'exemples positifs (resp. négatifs) parmi les exemples couverts,
$P_0^1 = \frac{n_0^1}{n_0}$; $P_0^0 = \frac{n_0^0}{n_0}$: Taux d'exemples positifs (resp. négatifs) parmi les exemples non-couverts.

La mesure d'entropie, définie précédemment, se réécrit, à l'aide des notations proposées, de la façon suivante :

$$H(P_1^1, P_1^0) = -P_1^1 \log_2 P_1^1 - P_1^0 \log_2 P_1^0$$

A partir de cette définition de l'entropie (dans le cas binaire), on définit les critères de performance suivants :

Gain d'Information	$IG(R) = H(P^1, P^0) - \sum_{i=0}^{1}(P_i \ H(P_i^1, P_i^0))$
Rapport de Gain	$GR(R) = \frac{IG(R)}{H(P_0, P_1)}$
Statistique de Goodman-Kruskal (*G Statistic*)	$G(R) = 2.N.IG(R).\ln 2$
Indice de Gini	$gini(R) = GI(P^1, P^0) - \sum_{i=0}^{1}(P_i \ GI(P_i^1, P_i^0))$ avec $GI(x, y) = 1 - (x^2 + y^2)$
χ^2	$\chi^2(R) = \frac{N \ (n_1^1 n_0^0 - n_0^1 n_1^0)^2}{n^1 n^0 n_1 n_0}$
Laplace	$L(R) = \begin{cases} \frac{n_1^1 + 1}{n_1^1 + n_1^0 + 2} & \text{si} \ \ n_1^1 \geq n_1^0 \\ \frac{n_1^0 + 1}{n_1^1 + n_1^0 + 2} & \text{si} \ \ n_1^1 < n_1^0 \end{cases}$

TAB. C.2 – Mesures d'évaluation de la performance d'une règle.

D

Expertise de quelques groupes de documents

Nous ne présenterons pas l'intégralité du rapport proposé par l'expert sur les résultats obtenus, mais uniquement les aspects les plus significatifs quant à notre objectif descriptif exploratoire.

D.1 Classification basée sur l'ensemble des 130 descripteurs

Nb. clusters	Nb. singletons	Coefficient de partition
58	14	1.32

De manière générale, les classes obtenues semblent reliées à la thématique des articles. Les singletons méritant une analyse spécifique, ils ont été étudiés séparément. Après analyse des groupes, 14 clusters ont été jugés thématiquement pertinents par l'expert sur un total de 44 groupes (après extraction des singletons), soit plus d'un quart des clusters.

De manière plus précise, l'expert nous apporte les observations suivantes :

• Les clusters 7 et 13 comprennent (uniquement) deux individus extraits de la même revue (textes 186 et 188 : Verbum 2001, *Sémantique des verbes, Nouvelles approches*; textes 167 et 168 : Verbum 2000, *Autour du futur*). Les thèmes affichés des deux numéros portent clairement sur un marqueur linguistique : le futur (effectivement élevé dans les textes 167 et 168), et les verbes (beaucoup d'infinitifs et de passifs infinitifs dans les textes 186 et 188).

• Les clusters 17, 34 et 36, constitués de trois à quatre individus comprennent chacun deux individus issus de la même revue.

• Les clusters 31 et 51 comprennent chacun trois articles provenant du même numéro thématique. Si l'on prend l'exemple du cluster 31, il est constitué de cinq individus, dont trois appartiennent à la même revue : Langue Française, 2001, *La Parole intérieure*, numéro thématique plutôt détonnant, souvent plus historique/épistémologique que linguistique, la parole intérieure étant difficile à appréhender de manière expérimentale. Les cinq textes présentent les points communs de contenir peu d'indices de formalisation, moins de ponctuations que les autres articles du corpus, très peu de renvois internes (pourtant très courant dans les textes contenant des exemples, *e.g.* "comme nous le voyons en (1),...''), et beaucoup de noms propres et de dates.

On voit cependant les limites de notre démarche eu égard au texte 126 (dans ce même cluster 31), traitant des scènes d'actions radiophoniques : le texte n'est clairement pas historique, contrairement aux quatre autres. Les noms propres ne renvoient pas nécessairement à des noms du champ, mais au « studio Charles Trenet » par exemple, tandis que les dates portent sur la diffusion des émissions radiophoniques.

• Le cluster 52, constitué de 8 individus comprend 4 articles extraits de cette même revue sur la parole intérieure.

• Le cluster 8 est particulièrement intéressant, dans la mesure où il comprend les 5 articles (sur un ensemble de 9 individus) d'un numéro thématique de langage traitant de la langue des signes ; la revue est particulière, dans la mesure où ses auteurs sont en majorité distincts, sociologiquement (enseignants dans le secondaire, enseignant de LSF, etc.).

• Les clusters 42, 48 et 53 sont thématiquement intéressants, bien que moins homogènes : le cluster 53 comprend 4 articles (sur un total de 5) d'un Verbum (Référence discursive et accessibilité cognitive), le cluster 42 quatre textes issus du Scolia de Reichler (Problèmes de sémantique et de relations entre micro- et macro-syntaxe) et le cluster 48, 3 textes issus du Scolia Contextes.

• Enfin, les classes 56 et 58, malgré leur nombre important d'individus semblent pertinentes : après vérification, le cluster 58 rassemble des articles portant sur l'objet verbal, et le cluster 56 contient des textes portant sur la sémantique/l'énonciation.

D.2 Étude des singletons

En premier lieu, on observe que la moitié des textes isolés proviennent de deux revues : LINX, qui est une revue universitaire thématique, et HEL qui est une revue de linguistique historique.

On retiendra d'abord HEL, dans la mesure où la revue est nettement moins représentée dans le corpus que LINX (10 articles *vs.* 28). Les textes publiés dans HEL sont effectivement spécifiques, entre histoire et linguistique. Les trois textes écartés sont ainsi faiblement structurés, contiennent davantage de dates et de noms propres que les autres textes, et deux textes sur trois recourent aux temps narratifs (passé simple, imparfait) dans le corps de l'article, ce qui est peu commun dans l'ensemble du corpus. Il semble, en outre, que les auteurs utilisent peu les structures impersonnelles en IL ou ON ; soit le texte comprend très peu de pronoms, soit il contient un nombre important de JE ou de NOUS.

Il n'est donc pas étonnant que ces textes aient été écartés ; leur appartenance au genre de l'article scientifique linguistique peut être questionnée.

Deux textes isolés de LINX sont extraits du même numéro thématique (*L'hypothétique*,

1999). Les deux textes relèvent par contre clairement du genre et du domaine, mais c'est leur objet et la méthodologie de traitement utilisée qui semble les avoir écartés.

Le premier texte (83) a visiblement été écarté du fait de l'emploi important des temps de l'hypothétique : conditionnels passé (2.33% pour 0.39% en moyenne), plus que parfait (2.13% pour 0.34%), conditionnels (1.94% pour 0.92%) et imparfaits (4.65% pour 1.42%). Ce texte présente également de nombreux pronoms de 1ère personne du singulier et seconde personne du pluriel, employés dans des exemples[1] : nombre important de JE (11.69% pour 2.99%) et de VOUS - emploi le plus élevé remarqué - (9.27% pour 0.40%).

Le second texte (86) privilégie les 1ère et 2nde personnes du singulier pour exemplifier l'objet étudié (« Si **tu** ne **m'**avais pas aidé, **je** n'aurais pas pu finir à temps »), ce qui explique la présence importante de JE (8.93% pour 2.99%) et TU (8.65% pour 0.54%). Le texte portant sur le chinois, on relève nettement moins de temps de l'hypothétique que dans le texte 83 ; le texte contient beaucoup d'infinitifs (28% pour 17.79%), liées aux nombreuses traductions (littérales) du chinois[2].

Les deux textes ont ainsi été écartés principalement en raison de leur objet, et plus spécifiquement des exemples donnés, instituant une relation 1ère/2nde personne.

La plupart des individus isolés semblent l'avoir été en raison du thème de l'article - généralement il s'agit d'un objet linguistique - et plus précisément en raison des nombreux exemples qu'ils contiennent, et qui agissent sur la morphosyntaxe du texte : ainsi, l'article de Francis Renaud (116), intitulé *Les bases de la quantification réciproque*, a été écarté en raison du nombre important de pronoms indéfinis (27.24% pour 2.83%) et réflexifs (26.26% pour 11.51%) quil contient ; celui de Manguin (210), intitulé *Construction d'espaces sémantiques associés aux verbes de déplacement d'objets à partir des données des dictionnaires informatisés des synonymes*, en raison des nombreux exemples de verbes à l'infinitif donnés (48.36% pour 17.79%) ; et celui de Manno (40), qui porte sur les actes « pseudo-directifs » de la communication écrite, en raison des nombreuses relations d'interlocution présentes dans les textes (valeurs très élevées pour les marqueurs associés aux pronoms JE-TU-NOUS-VOUS).

A noter que les textes écartés détenaient, la plupart du temps, des « records » d'emplois de certaines variables, et que deux textes proviennent du même auteur (Crévenat-Werner). On peut penser que leur isolement est dû à des critères stylométriques.

D.3 Classifications basées sur les autres ensembles de descripteurs

Les schémas de clustering obtenus à partir des autres tagsets semblent avoir été constitués à partir de critères davantage stylométriques et génériques, que proprement thématiques : par exemple, les textes 9 et 128 sont tous deux formels et descriptifs, et recourent au futur descriptif normatif (*e.g.* "le SN sera constitué de deux éléments [...]"). Les objets scientifiques diffèrent, mais la méthodologie d'analyse est similaire, ce qui explique leur regroupement.

Les textes entiers (full text) ont été soumis à l'étiquetage, à l'exception des bibliographies et des résumés, s'il y en a. On voit nettement l'intérêt d'une extraction et d'un traitement des composantes de l'article lorsqu'on examine les regroupements de textes

[1] « si **j'**étais à ta place, **je** ferais réparer cette machine au plus vite » ; « **vous** avez donc dix doigts », etc.

[2] e.g. « anshi wancheng zhei jian gongzuo / à temps finir ce Cl. Travail ».

Type de variables	Nb. clusters	Nb. singletons	Coefficient de partition
Temps/Personnes	45	6	1.32
Discours scientifique	51	9	1.25

fondés sur des similarités morphosyntaxiques, dues à des composantes différentes : ainsi, les textes 45 et 80 ont une répartition des temps verbaux très similaires, mais ces caractéristiques sont liées aux exemples dans le texte de Feuillet, et dans le corps de l'article dans celui de Ildefonse, *e.g.* usage de l'imparfait, dans les exemples chez Feuillet (« si j'en avais, je t'en donnerais »), et dans le texte chez Ildefonse (« Pouvait-on prétendre qu'Apollonius voyait la relation de l'accusatif [...]»).

Bibliographie

[1] Aas (K.) et Eikvil (L.). – *Text Categorisation : A Survey.* – Rapport technique, Norwegian Computing Center, 1999.

[2] Agarwal (R.). – (Almost) automatic semantic feature extraction from technical text. *In : Proceedings of the human language technology workshop.* pp. 378–383. – Princeton, New Jersey, 1994.

[3] Agirre (E.), Ansa (O.), Hovy (E.) et Martinez (D.). – Enriching Very Large Ontologies Using the WWW. *In : Proceedings of the Ontology Learning Workshop, ECAI.* – Berlin, Germany, 2000.

[4] Agrawal (R.), Gehrke (J.), Gunopulos (D.) et Raghavan (P.). – Automatic subspace clustering of high dimensional data for data mining applications. *In : Proceedings of Int. Conf. Management of Data ACM-SIGMOD*, pp. 94–105. – Seattle, 1998.

[5] Ankerst (M.), Breunig (M.), Kriegel (H. P.) et Sander (J.). – OPTICS : Ordering Points To Identify the Clustering Structure. *In : Proceedings of Int. Conf. Management of Data ACM-SIGMOD*, pp. 49–60. – Philadelphia, Pennsylvania, USA, 1999.

[6] Apté (C.), Damerau (F.) et Weiss (S. M.). – Automated learning of decision rules for text categorization. *ACM Trans. Inf. Syst.*, vol.12, N° 3, 1994, pp. 233–251.

[7] Baker (L. D.) et McCallum (A. K.). – Distributional clustering of words for text classification. *In : Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval.* pp. 96–103. – Melbourne, AU, 1998.

[8] Ball (G. H.) et Hall (D. J.). – *A novel method of data analysis and pattern classification.* – Rapport technique, Menlo Park, CA, Stanford Research Insitute, 1965.

[9] Baraldi (A.) et Blonda (P.). – A survey of fuzzy clustering algorithms for pattern recognition. II. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol.29, 1999, pp. 786–801.

[10] Berkhin (P.). – *Survey Of Clustering Data Mining Techniques.* – Rapport technique, San Jose, CA, Accrue Software, 2002.

[11] Bezdek (J. C.). – Pattern Recognition with Fuzzy Objective Function Algoritms. *Plenum Press, New York*, 1981.

[12] Biernacki (C.), Celeux (G.), Govaert (G.), Langrognet (F.) et Vernaz (Y.). – Mixmod : High Performance Model-Based Cluster and Discriminant Analysis. *http ://www-math.univ-fcomte.fr/mixmod/index.php*, 2001.

[13] Bishop (C.M.). – *Neural Networks for Pattern Recognition.* – Oxford University Press, 1995.

[14] Bisson (G.). – Learning in FOL with a similarity measure. *In : 11th National Conf. on Artificial Intelligence (AAAI).* pp. 82–87. – San Jose, CA, 1992.

[15] Bisson (G.). – La similarité : une notion symbolique/numérique. *Apprentissage symbolique-numérique (tome 2)*, 2000.

[16] Boley (D.). – Principal Direction Divisive Partitioning. *Data Mining and Knowledge Discovery*, vol.2, N° 4, 1998, pp. 325–344.

[17] Bomze (I.), Budinich (M.), Pardalos (P.) et Pelillo (M.). – The maximum clique problem. *In : Handbook of Combinatorial Optimization*, éd. par Du (D.-Z.) et Pardalos (P. M.). – Kluwer Academic Publishers, Boston, MA, 1999.

[18] Brandenburg (F.), Edachery (J.) et Sen (A.). – Graph clustering using distance-k cliques. *Lecture Notes in Computer Sciences*, vol.1731, 1999, pp. 98–106.

[19] Brézellec (P.) et Didier (G.). – GIZMO : un algorithme de grille cherchant des clusters homogènes. *In : Conférence francophone d'Apprentissage (CAp'2001)*, pp. 101–116. – Grenoble, France, Juin 2001.

[20] Brito (P.). – Symbolic Pyramidal Clustering. *In : Proceedings of the Indo-French Workshop on Symbolic Data Analysis and its Applications.* – Paris, France, September 1997.

[21] Cheeseman (P.) et Stutz (J.). – Bayesian Classification (AutoClass) : Theory and Results. *In : Advances in Knowledge Discovery and Data Mining*, pp. 153–180. – 1996.

[22] Chuai-Aree (S.), Lursinsap (C.), Sophatsathit (P.) et Siripant (S.). – Fuzzy C-Mean : A statistical feature classification of text and image segmentation method. *In : Proc. of Intern. Conf. on Intelligent Technology*, pp. 279–284. – Assumption University Bangkok, Thailand, December 2000.

[23] Church (K. W.) et Hanks (P.). – Word association norms, mutual information, and Lexicography. *In : Proceedings of the 27th. Annual Meeting of the Association for Computational Linguistics.* pp. 76–83. – Vancouver, B.C., 1989.

[24] Clark (P.) et Niblett (T.). – The CN2 induction algorithm. *Machine Learning*, vol.3, 1989, pp. 261–284.

[25] Clavier (V.), Cleuziou (G.) et Martin (L.). – Organisation conceptuelle de mots pour la recherche d'information sur le web. *In : Conférence d'Apprentissage CAp'2002*, éd. par PUG Presses Universitaires de Grenoble, pp. 220–235. – Orléans, Juin 2002.

[26] Cleuziou (G.). – Regroupements non-disjoints de mots pour la classification de documents. *In : Première Conférence en Recherche d'Information et Applications CORIA'2004.* pp. 41–56. – toulouse, janvier 2004.

[27] Cleuziou (G.), Clavier (V.) et Martin (L.). – Une méthode de regroupement de mots fondée sur la recherche de cliques dans un graphe de cooccurrences. *In : 5èmes rencontres Terminologie et Intelligence Artificielle*, éd. par ENSAIS (LIIA), pp. 179–182. – Strasbourg, France, Mars 2003. Poster.

[28] Cleuziou (G.), Martin (L.), Clavier (V.) et Vrain (C.). – DDOC : Overlapping Clustering of Words for Document Classification. *In : 11th Symposium on String Processing and Information Retrieval*, éd. par LNCS. pp. 127–128. – Padova, Italy, October 2004.

[29] Cleuziou (G.), Martin (L.) et Vrain (C.). – Disjunctive Learning with a Soft-Clustering Method. *In : 13th International Conference on Inductive Logic Programming*, éd. par LNCS. pp. 75–92. – Szeged, Hungary, September 2003.

[30] Cleuziou (G.), Martin (L.) et Vrain (C.). – PoBOC : un algorithme de "soft-clustering". Applications à l'apprentissage de règles et au traitement de données textuelles. *In : Journées Francophones d' Extraction et de Gestion des Connaissances EGC'2004*. pp. 217–228. – Clermont-Ferrand, janvier 2004.

[31] Cleuziou (G.), Martin (L.) et Vrain (C.). – PoBOC : an Overlapping Clustering Algorithm. Application to Rule-Based Classification and Textual Data. *In : Proceedings of the 16th European Conference on Artificial Intelligence*, éd. par R. López de Mántaras and L. Saitta, IOS Press, pp. 440–444. – Valencia, Spain, August 22-27 2004.

[32] Cohen (W.) et Hirsh (H.). – Joins that generalize : text classification using WHIRL. *In : Proceedings of KDD-98, 4th International Conference on Knowledge Discovery and Data Mining*, éd. par Rakesh Agrawal and Paul E. Stolorz and Gregory Piatetsky-Shapiro. pp. 169–173. – New York, US, 1998.

[33] Dagan (I.), Karov (Y.) et Roth (D.). – Mistake-driven learning in text categorization. *In : Proceedings of EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing*, éd. par Claire Cardie and Ralph Weischedel. pp. 55–63. – Providence, US, 1997.

[34] Dash (M.), Choi (K.), Scheuermann (P.) et Liu (H.). – Feature selection for clustering - A Filter Solution. *In : proceedings of IEEE International Conference of Data Mining (ICDM 2002)*. – Maebashi City, Japan, December 2002.

[35] Datta (P.) et Kibler (D.). – Symbolic Nearest Mean Classifiers. *In : proceedings of Fifteenth National Conference on Artificial Intelligence*. pp. 82–87. – Providence, Rhode Island, 1997.

[36] Davies (D. L.) et Bouldin (D. W.). – A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.1, N° 2, 1979, pp. 224–227.

[37] Debourges (I.). – Construction de cartes pour l'exploration de corpus. – Thèse de doctorat. LIFO, Université d'Orléans, juillet 2002.

[38] Deerwester (S. C.), Dumais (S. T.), Landauer (T. K.), Furnas (G. W.) et Harshman (R. A.). – Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, vol.41, N° 6, 1990, pp. 391–407.

[39] DeJong (K. A.), Spears (W. M.) et Gordon (D. F.). – Using Genetic Algorithms for Concept Learning. *special issue on genetic algorithms for Machine Learning Journal*, vol.13, 1993, pp. 161–188.

[40] Dempster (A.P.), Laird (N.M.) et Rubin (D.B.). – Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of Royal Statistical Society B*, vol.39, 1977, pp. 1–38.

[41] DeRaedt (L.), Lavrac (N.) et Dzeroski (S.). – Multiple Predicate Learning. *In : Proceedings of the Thirteen International Joint Conference on Artificial Intelligence*. pp. 1037–1043. – Chambéry, France, 1993.

[42] Dhillon (I. S.), Mallela (S.) et Kumar (R.). – A divisive information theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Researches*, vol.3, 2003, pp. 1265–1287.

[43] Dice (L. R.). – Measures of the amount of ecologic associations between species. *Journal of Ecology*, 1945.

[44] Diday (E.). – Optimisation en classification automatique et reconnaissance de formes. *Note Scient. INRIA n ° 6*, 1972.

[45] Diday (E.). – La méthode des nuées dynamiques. *Rev. Stat. Appliquées*, vol.XIX, N° 2, 1975, pp. 19–34.

[46] Diday (E.). – *Une représentation visuelle des classes empiétantes : Les Pyramides.* – Rapport technique, INRIA n ° 291, Rocquencourt 78150, France, 1984.

[47] Diday (E.) et Esposito (F.). – An introduction to Symbolic Data Analysis and the Sodas Software. *International Journal on Intelligent Data Analysis*, vol.7, N° 6, 2003.

[48] Domingos (P.). – The RISE System : Conquering Without Separating. *In : Sixth IEEE International Conference on Tools with Artificial Intelligence*, pp. 704–707. – New Orleans, LA, 1994.

[49] Domingos (P.) et Pazzani (M. J.). – Beyond Independence : Conditions for the Optimality of the Simple Bayesian Classifier. *In : International Conference on Machine Learning*, pp. 105–112. – Bari, Italy, 1996.

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

[51] Dunn (J. C.). – Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, vol.4, 1974, pp. 95–104.

[52] Emde (W.) et Wettschereck (D.). – Relational Instance-Based Learning. *In : 13th Int. Conf. on Machine Learning (ICML'96)*, éd. par L. (Saitta). pp. 122–130. – Bari, Italy, 1996.

[53] Enguehard (C.). – Apprentissage naturel automatique d'un réseau sémantique. – Thèse de doctorat. Université de Compiègne, 1992.

[54] Eschrich (S.). – *Hierarchical Fuzzy Clustering as a Top-Down Perceptual Organization Process.* – Rapport technique, Computer Vision Project, Final Report, 2001.

[55] Ester (M.), Kriegel (H. P.), Sander (J.), Wimmer (M.) et Xu (X.). – Density-Connected Sets and their Application for Trend Detection in Spatial databases. *In : Third International Conference on Knowledge Discovery and Data Mining.* – Newport Beach, California, USA, 1997.

[56] Ester (M.), Kriegel (H. P.), Sander (J.) et Xu (X.). – A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *In : Second International Conference on Knowledge Discovery and Data Mining*, éd. par Simoudis (E.), Han (J.) et Fayyad (U.). pp. 226–231. – Portland, Oregon, 1996.

[57] Estivill-Castro (V.), Lee (I.) et Murray (A. T.). – Criteria on Proximity Graphs for Boundary Extraction and Spatial Clustering. *In : Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining.* pp. 348–357. – Springer-Verlag.

[58] Fano (R.M.). – *Transmission of Information : A Statistical Theory of Communications.* – Cambridge, Mass., MIT Press, 1961.

[59] Faure (D.) et Nédellec (C.). – ASIUM : Learning subcategorization frames and restrictions of selection. *In : 10th European Conference on Machine Learning (ECML 98) - Workshop on Text Mining*, éd. par Y. Kodratoff. – Chemnitz, Germany, 1998.

[60] Fisher (D.). – Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, vol.2, 1987, pp. 139–172.

[61] Fraley (C.) et Raftery (A.). – Mclust : Software for modelbased cluster analysis. *Journal of Classification*, vol.16, 1999, pp. 297–306.

[62] Fuhr (N.) et Buckley (C.). – A probabilistic learning approach for document indexing. *ACM Trans. Inf. Syst.*, vol.9, N° 3, 1991, pp. 223–248.

[63] Fukuyama (Y.) et Sugeno (M.). – A New Method of Choosing the Number of Clusters for the Fuzzy C-means Method. *In : Proc. 5th Fuzzy Syst. Symp.*, pp. 247–250. – Japan, 1989.

[64] Fung (G.). – A comprehensive overview of basic clustering algorithms. 2001.

[65] Gaume (B.), Hathout (N.) et Muller (P.). – Désambiguïsation par proximité structurelle. *In : Actes de la Conférence Traitement Automatique du Langage Naturel (TALN'2004)*. – Fez, Maroc, 2004.

[66] Gennari (J. H.). – *An experimental study of concept formation*. – Technical Report nN° 90-06, Irvine : University of California, Department of Information and Computer Science, 1990.

[67] Gennari (J. H.), Langley (P.) et Fisher (D.). – Models of incremental concept formation. *Journal of Artificial Intelligence*, vol.40, 1989, pp. 11–61.

[68] Gil (À. J.), Capdevila (C.) et Arcas (A.). – On the Efficiency and Sensitivity of a Pyramidal Classification Algorithm. *Journal of Economic Literature*, 1994.

[69] Gluck (M. A.) et Corter (J. E.). – Information, Uncertainty and the Utility of Categories. *In : Proceedings of the Seventh Annual Conference of the Cognitive Science Society*. pp. 283–287. – Irvine, California, 1985.

[70] Gnanadesikan (R.), Kettenring (J. R.) et Landwehr (J.M.). – Interpreting and assessing the results of cluster analyses. *Bulletin of the International Statistical Institute*, vol.47, N° 2, 1977, pp. 451–463.

[71] Grabmeier (J.) et Rudolph (A.). – Techniques of Cluster Algorithms in Data Mining. *Data Mining and Knowledge Discovery*, nN° 6, 2002, pp. 303–360.

[72] Grefenstette (G.). – *Explorations in automatic thesaurus discovery*. – Kluwer Academic Pub., 1994.

[73] Grefenstette (G.). – The WWW as a Resource for Example-Based MT Tasks. *In : Proc. ASLIB Translating and the Computer 21 Conf.* – London, England, 1999.

[74] Guha (S.), Rastogi (R.) et Shim (K.). – CURE : an efficient clustering algorithm for large databases. *In : proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 73–84. – Seattle, Washington, 1998.

[75] Guha (S.), Rastogi (R.) et Shim (K.). – ROCK : A Robust Clustering Algorithm for Categorical Attributes. *Information Systems*, vol.25, N° 5, 2000, pp. 345–366.

[76] Halkidi (M.), Batistakis (Y.) et Vazirgiannis (M.). – Clustering Validity Checking Methods : Part II. *ACM SIGMOD*, vol.31, N° 3, 2002, pp. 19–27.

[77] Halkidi (M.) et Vazirgiannis (M.). – Clustering Validity Assessment : Finding the Optimal Partitioning of a Data Set. *In : proceedings of IEEE International Conference of Data Mining (ICDM 2001)*, pp. 187–194. – San Jose, California, USA, 2001.

[78] Harris (Z.), Gottfried (M.), Ryckman (T.), Mattick (P.), Daladier (A.), Harris (T. N.) et Harris (S.). – *The form of Information in Science : Analysis of an immunology sublanguage*. – Dordrecht : Kluwer Adademic Publishers, 1989.

[79] Hathaway (R. J.), Davenport (J. W.) et Bezdek (J. C.). – Relational duals of the c-means clustering algorithms. *Pattern Recognition*, vol.22, N° 2, 1989, pp. 205–212.

[80] Hawkins (D.). – *Identification of Outliers*. – London, Chapman and Hall, 1980.

[81] Hertz (J.), Krogh (A.) et Palmer (R. G.). – *An Introduction to the Theory of Neural Computation*. – Santa Fe Institute Studies in the Sciences of Complexity lecture notes. Addison-Wesley Longman Publ. Co., Inc., Reading, MA., 1991.

[82] Hinneburg (A.) et Keim (D. A.). – An Efficient Approach to Clustering in Large Multimedia Databases with Noise. *In : Knowledge Discovery and Data Mining*, pp. 58–65. – New York City, USA, 1998.

[83] Hodson (F. R.), Sneath (P. H. A.) et Doran (J. E.). – Some experiments in the numerical analysis of archaeological data. *Biometrika*, vol.53, 1966, pp. 311–324.

[84] Hoff (W. A.), Michalski (R. S.) et Stepp (Robert E.). – *INDUCE 2 : A program for learning structural descriptions from examples*. – Rapport technique nN° 904, Urbana, Illinois, Department of Computer Science, University of Illinois at Urbana-Champaign, 1983.

[85] Hofmann (T.). – Probabilistic Latent Semantic Indexing. *In : Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pp. 50–57. – Berkeley, California, August 1999.

[86] Holman (E. W.). – Evolutionary and psychological effects in pre-evolutionary classifications. *Journal of Classification*, vol.2, 1985, pp. 29–39.

[87] Hull (D.). – Improving text retrieval for the routing problem using latent semantic indexing. *In : Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. pp. 282–291. – Dublin, Ireland, 1994.

[88] Hösel (V.) et Walcher (S.). – *Clustering Techniques : A Brief Survey*. – Rapport technique, AMS Subject Classication : 62H30, 68T10, 62-07 1, 2000.

[89] Ibekwe-San Juan (F.). – Recherche des tendances thématiques dans les publications scientifiques. définition d'une méthodologie fondée sur la linguistique. – Thèse de doctorat. Université de Grenoble3, 1997.

[90] Jaccard (P.). – The distribution of the flora in the alpine zone. *New Phytol.*, vol.11, 1912, pp. 37–50.

[91] Jain (A. K.), Murty (M. N.) et Flynn (P. J.). – Data clustering : a review. *ACM Computing Surveys*, vol.31, N° 3, 1999, pp. 264–323.

[92] Jaromczyk (J.W.) et Toussaint (G.T.). – Relative Neighborhood Graphs And Their Relatives. *P-IEEE*, vol.80, 1992, pp. 1502–1517.

[93] Joachims (T.). – Text categorization with support vector machines : learning with many relevant features. *In : Proceedings of ECML-98, 10th European Conference*

on Machine Learning, éd. par Claire Nédellec and Céline Rouveirol. pp. 137–142. – Chemnitz, DE, 1998.

[94] Johnson (S. C.). – Hierarchical clustering schemes. *Psychometrika*, vol.32, 1967, pp. 241–254.

[95] Karypis (G.), Han (E.-H.) et Kumar (V.). – CHAMELEON : Hierarchical Clustering Using Dynamic Modeling. *Computer*, vol.32, N° 8, 1999, pp. 68–75.

[96] Kaufman (L.) et Rousseeuw (P. J.). – Clustering by means of medoids. *In Dodge, Y. (Ed.) Statistical Data Analysis based on the L1 Norm*, 1987, pp. 405–416.

[97] Kaufman (L.) et Rousseeuw (P. J.). – *Finding Groups in Data. An Introduction to Cluster Analysis.* – John Wiley & Sons, Inc., 1990.

[98] Kearns (M.), Mansour (Y.) et Ng (A.). – An Information-Theoretic Analysis of Hard and Soft Assignment Methods for Clustering. *In : Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97).* pp. 282–293. – San Francisco, CA, 1997.

[99] Kettenring (J. R.), Rogers (W. H.), Smith (M. E.) et Warner (J. L.). – Cluster analysis applied to the validation of course objectives. *J. Educ. Statist.*, vol.1, 1976, pp. 39–57.

[100] Knorr (E. M.) et Ng (R. T.). – Algorithms for Mining Distance-Based Outliers in Large Datasets. *In : Proceedings of the 24rd International Conference on Very Large Data Bases*. pp. 392–403. – New York City, USA, 1998.

[101] Kohonen (T.). – Self-Organization and Associative Memory. *Springer*, 1984.

[102] Krishnapuram (R.), Joshi (A.) et Yi (L.). – A fuzzy relative of the k-medoids algorithm with application to document and snippet clustering. *In : Proc. IEEE Intl. Conf. Fuzzy Systems*, pp. 1281–1286. – Korea, 1999.

[103] Lallich-Boidin (G.). – Communication homme-machine et recherche d'information fondée sur le traitement automatique des langues. Applications, bilan, perspectives. – Mémoire d'habilitation à diriger des recherches. Université Stendhal, Grenoble3, 1998.

[104] Lang (K.). – NewsWeeder : learning to filter netnews. *In : Proceedings of the 12th International Conference on Machine Learning*. pp. 331–339. – San Mateo, CA, USA, 1995.

[105] Larkey (L. S.) et Croft (W. B.). – Combining classifiers in text categorization. *In : Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, éd. par Frei (H.-P.), Harman (D.), Schäuble (P.) et Wilkinson (R.). pp. 289–297. – Zürich, CH, 1996.

[106] Law (M.), Figueiredo (M.) et Jain (A. K.). – Feature selection in mixture-based clustering. *In : Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pp. 609–616. – Vancouver, Canada, 2002.

[107] Lee (D. H.) et Kim (M. H.). – Database summarization using fuzzy isa hierarchies. *Ieee Trans. On Systems Man And Cybernetics Part B- Cybernetics*, vol.27, 1997, pp. 68–78.

[108] Lee (D.T.) et Schachter (B.J.). – Two Algorithms for Constructing a Delaunay Triangulation. *Int. J. Computer Information Sci.*, vol.9, 1980, pp. 219–242.

[109] Lefèvre (P.). – *La Recherche d'informations. Du texte intégral au thésaurus.* – Paris, Hermès, 2000.

[110] Lelu (A.). – *Modèles neuronaux pour l'analyse de données documentaires et textuelles*. – Thèse de doctorat, Université de Paris VI, mars 1993.

[111] Lewis (D. D.). – An evaluation of phrasal and clustered representations on a text categorization task. *In : Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*. pp. 37–50. – Copenhagen, Denmark, 1992.

[112] Li (Y. H.) et Jain (A. K.). – Classification of text documents. *Comput. J.*, vol.41, N° 8, 1998, pp. 537–546.

[113] Lin (D.). – An Information-Theoretic Definition of Similarity. *In : Proceedings of the Fifteenth International Conference on Machine Learning*. pp. 296–304. – Madison, Wisconsin USA, 1998.

[114] Lin (D.) et Pantel (P.). – Induction of semantic classes from natural language text. *In : Proceedings of SIGKDD*, pp. 317–322. – San Francisco, CA, 2001.

[115] Lin (K. I.) et Kondadadi (R.). – A Word-Based Soft Clustering Algorithm for Documents. *In : Proceedings of 16th International Conference on Computers and Their Applications*. – Seattle, Washington, 2001.

[116] Liu (Huan) et Setiono (Rudy). – A Probabilistic Approach to Feature Selection - A Filter Solution. *In : International Conference on Machine Learning*, pp. 319–327. – Bari, Italy, 1996.

[117] MacQueen (J.). – Some methods for classification and analysis of multivariate observations. *In : Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability*. pp. 281–297. – Berkeley, 1967.

[118] Malerba (D.), Esposito (F.), Gioviale (V.) et Tamma (V.). – Comparing Dissimilarity Measures for Symbolic Data Analysis. *In : Int. Conferences on Exchange of Technologies and Know-How and New Techniques & Technologies for Statistics*. – Crete, Greece, 2001.

[119] Mallat (S.). – A wavelet tour of signal processing. *Academic Press*, 1998.

[120] Martin (L.) et Moal (F.). – A Language-Based Similarity Measure. *In : 12th European Conference on Machine Learning ECML 2001*. pp. 336–347. – Freiburg, Germany, 2001.

[121] Matula (D.W.) et Sokal (R.R.). – Properties of Gabriel graphs relevant to geographic variation research and clustering of points in the plane. *Geogr. Anal.*, vol.12, N° 3, 1980, pp. 205–222.

[122] Merz (C.J.) et Murphy (P.M.). – UCI repository of machine learning databases. 1998.

[123] Michalski (R. S.). – On the quasi-minimal solution of the general covering problem. *In : V International Symposium on Information Processing (FCIP 69)*, pp. 125–128. – Yugoslavia, 1969.

[124] Michalski (R. S.). – *Knowledge acquisition through conceptual clustering : A theoretical framework and an algorithm for partitioning data into conjunctive concepts.* – Technical Report nN° 1026, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1980.

[125] Michalski (R. S.), Stepp (R. E.) et Diday (E.). – A recent advance in data analysis : Clustering objects into classes characterized by conjuctive concepts. *Pattern Recognition*, vol.1, 1983.

[126] Mitchell (T.). – *Machine Learning*. – New York, McGraw Hill, 1997.

[127] Mitra (P.), Murthy (C.A.) et Pal (S.K.). – Unsupervised Feature Selection Using Feature Similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.24, N° 4, 2002.

[128] Moody (J.) et Darken (C.). – Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, vol.1, 1989, pp. 281–294.

[129] Morin (E.). – Extraction de liens sémantiques entre termes à partir de corpus de textes techniques. – Thèse de doctorat. IRIN, Université de Nantes, décembre 1999.

[130] Moulinier (I.) et Ganascia (J.G.). – Applying an existing machine learning algorithm to text categorization. *In : Connectionist, statistical, and symbolic approaches to learning for natural language processing, workshop in IJCAI '95*, éd. par Stefan Wermter and Ellen Riloff and Gabriele Scheler. pp. 343–354. – Montreal, Canada, 1996.

[131] Muggleton (S.). – Inverting Entailment and Progol. *New Generation Computing*, vol.13, 1995, pp. 245–286.

[132] Muggleton (S.) et Feng (C.). – Efficient induction of logic programs. *In : Proceedings of the 1st Conference on Algorithmic Learning Theory*. pp. 368–381. – Tokyo, Japan, 1990.

[133] Muller (C.), Polanco (X.), Royaute (J.) et Toussaint (Y.). – *Acquisition et structuration des connaissances en corpus : éléments méthodologiques*. – Technical Report nN° RR-3198, Inria, Institut National de Recherche en Informatique et en Automatique, Juin 1997.

[134] Murty (M. N.), Babu (T. R.) et Agrawal (V. K.). – Clustering Large Symbolic Datasets. *In : International Workshop on Symbolic Data Analysis*. – Paris, France, 2004.

[135] Nadif (M.) et Jollois (F. X.). – Accélération de EM pour données qualitatives : étude comparative de différentes versions. *In : EGC 2004, 4th French-Speaking Conference on Knowledge Discovery and Knowledge Management*. pp. 253–264. – Clermont-Ferrand, France, Janvier 2004.

[136] Ng (T. R.) et Han (J.). – Efficient and Effective Clustering Methods for Spatial Data Mining. *In : Proceedings of 20th International Conference on Very Large Data Bases VLDB'94*, éd. par Bocca (J. B.), Jarke (M.) et Zaniolo (C.). pp. 144–155. – Santiago de Chile, Chile, 1994.

[137] Pantel (P.). – *Clustering by Committee*. – Ph.d. dissertation, Department of Computing Science, University of Alberta, 2003.

[138] Pantel (P.) et Lin (D.). – Discovering word senses from text. *In : Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 613–619. – Edmonton, Alberta, Canada, 2002.

[139] Paterson (M.S.) et Yao (F.F.). – On Nearest Neighbor Graphs. *Automata, Languages and Programming*, vol.623, 1992, pp. 416–426.

[140] Peel (D.) et McLachlan (G.). – *User's guide to EMMIX - version*. – Technical report, University of Queensland, Australia, 1998.

[141] Pellegrini (F.). – Static mapping by dual recursive bipartitioning of process and architecture graphs. *IEEE*, 1994, pp. 486–493.

[142] Pereira (F. C. N.), Tishby (N.) et Lee (L.). – Distributional Clustering of English Words. *In : 31st Annual Meeting of the Association for Computational Linguistics*, pp. 183–190. – Ohio, USA, 1993.

[143] Pessiot (J. F.), Caillet (M.), Amini (M. R.) et Gallinari (P.). – Apprentissage non-supervisé pour la segmentation automatique de textes. *In : Première Conférence en Recherche d'Information et Applications CORIA'2004*, pp. 213–227. – Toulouse, France, janvier 2004.

[144] Plotkin (G. D.). – A note on inductive generalization. *Machine Intelligence*, vol.5, 1970.

[145] Poudat (C.) et Cleuziou (G.). – Genre and Domain Processing in an Information Retrieval Perspective. *In : Third International Conference on Web Engineering*, éd. par LNCS. pp. 399–402. – Oviedo, Spain, 2003.

[146] Prié (Y.). – Sur la piste de l'indexation conceptuelle de documents : une approche par l'annotation. *L'indexation, Document Numérique*, vol.4, N° 1-2, 2000, pp. 11–35.

[147] Qiu (Y.) et Frei (H. P.). – Concept-based query expansion. *In : Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, pp. 160–169. – Pittsburgh, US, 1993.

[148] Quinlan (J. R.). – Induction of decision trees. *Machine Learning*, vol.1, 1986, pp. 81–106.

[149] Quinlan (J. R.). – *C4.5 : Programs for Machine Learning*. – Morgan Kaufmann, 1993.

[150] Quinlan (J. R.) et Cameron-Jones (R. M.). – Induction of logic programs : FOIL and related systems. *New Generation Computing, Special issue on Inductive Logic Programming*, vol.13, N° 3-4, 1995, pp. 287–312.

[151] Rabiner (L. R.), Levinson (S. E.), Rosenberg (A. E.) et Wilpon (J. G.). – Speaker independent recognition of isolated words using clustering techniques. *IEEE Trans. Accoust. Speech Signal Process.*, vol.27, 1979, pp. 336–349.

[152] Rand (W. M.). – Objective Criteria for the evaluation of Clustering Methods. *Journal of the American Statistical Association*, vol.66, 1971, pp. 846–850.

[153] Rao (C. R.). – The utilization of multiple measurements in problems of biological classification. *Journal of Royal Statistical Society B*, vol.10, 1948, pp. 159–203.

[154] Resnik (P.). – Using Information Content to Evaluate Semantic Similarity in a Taxonomy. *In : Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 95*, pp. 448–453. – Montréal, Canada, 1995.

[155] Riloff (E.) et Shepherd (J.). – A corpus-based approach for building semantic lexicons. *In : Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pp. 117–124. – Somerset, New Jersey, 1997.

[156] Robertson (S. E.) et Harding (P.). – Probabilistic automatic indexing by learning from human indexers. *J. Document 40*, vol.4, 1984, pp. 264–270.

[157] Rodríguez (O.) et Diday (E.). – Pyramidal Clustering Algorithms in ISO-3D Project. *In : Workshop on Symbolic Data Analysis (PKDD 2000)*. – Lyon, France, 2000.

[158] Ruge (G.), Schwarz (C.) et Warner (A. J.). – Effectiveness and Efficiency in Natural Language Processing for Large Amounts of Text. *Journal of the American Society for Information Science*, vol.6, N° 42, 1991, pp. 450–456.

[159] Salton (G.) et McGill (M. J.). – *Introduction to Modern Information Retrieval*. – McGraw-Hill, Inc., 1986.

[160] Sander (J.), Ester (M.), Kriegel (H. P.) et Xu (X.). – Density-Based Clustering in Spatial Databases : The Algorithm GDBSCAN and its Applications. *Data Mining and Knowledge Discovery*, vol.2, 1998, pp. 169–194.

[161] Schütze (H.), Hull (D. A.) et Pedersen (J. O.). – A Comparison of Classifiers and Document Representations for the Routing Problem. *In : Research and Development in Information Retrieval*, pp. 229–237. – Seattle, Washington, USA, 1995.

[162] Sebag (M.). – Distance Induction in First Order Logic. *In : Proceedings of ILP'97*. pp. 264–272. – Prague, Czech Republic, 1997.

[163] Sebag (M.) et Schoenauer (M.). – *Topics in Case-Based Reasonning*, chap. A Rule-based Similarity Measure, pp. 119–130. – Springer-Verlag, 1994, *LNAI*, volume 837.

[164] Sebastiani (F.). – Machine learning in automated text categorization. *ACM Comput. Surv.*, vol.34, N° 1, 2002, pp. 1–47.

[165] Sheikholeslami (G.), Chatterjee (S.) et Zhang (A.). – WaveCluster : A Multi-Resolution Clustering Approach for Very Large Spatial Databases. *In : Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pp. 428–439. – New York City, USA, 1998.

[166] Shin (S. I.) et Choi (K. S.). – Automatic Word Sense Clustering Using Collocation for Sense Adaptation. *In : The Second Global Wordnet Conference (GWC'04)*. pp. 320–325. – Brno, Czech Republic, 2004.

[167] Siegel (J. H.), Goldwyn (R. M.) et Friedman (H.P.). – Pattern and process of the evolution of human septic shock. *Surgery*, vol.70, 1971, pp. 232–245.

[168] Slonim (N.) et Tishby (N.). – Document clustering using word clusters via the information bottleneck method. *In : Research and Development in Information Retrieval*, pp. 208–215. – Athenes, Greece, 2000.

[169] Smadja (F. Z.). – Retrieving Collocations from Text : Xtract. *Computational Linguistics*, vol.19(1), 1994, pp. 143–177.

[170] Smadja (F. Z.), McKeown (K.) et Hatzivassiloglou (V.). – Translating collocations for bilingual lexicons : A statistical approach. *Computational Linguistics*, vol.1, N° 22, 1996, pp. 1–38.

[171] Sneath (P. H. A.) et Sokal (R. R.). – Numerical Taxonomy - The Principles and Practice of Numerical Classification. – San Francisco, W. H. Freeman and Compagny, 1973.

[172] Sokal (R. R.) et Michener (C. D.). – A Statistical Method for Evaluating Systematic Relationships. *University of Kansas Science Bulletin*, vol.38, 1958, pp. 1409–1438.

[173] Sorensen (T.). – A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons. *Biologiske Skrifter*, vol.5, 1948, pp. 1–34.

[174] Sowa (J. F.). – Conceptual structures : Information processing in mind and machine. – Addison-Wesley, 1984.

[175] Sparck-Jones (K.). – *Synonymy and Semantic Classification*. – Edinburg University Press, 1987.

[176] Srinivasan (A.), Muggleton (S.), King (R. D.) et Sternberg (M. J. E.). – Mutagenesis : ILP experiments in a non-determinate biological domain. *In : Proceedings of the Fourth Inductive Logic Programming Workshop*. – Bon, Germany, 1994.

[177] Stepp (R. E.) et Michalski (R. S.). – Conceptual Clustering : Inventing Goal-Oriented Classifications of Structured Objects. *In : Machine Learning : An A.I. Approach*, éd. par Michalski (R.), Carbonell (J.) et Mitchell (T.). – Kaufmann, 1986.

[178] Toussaint (G.T.). – Algorithms for computing relative neighborhood graph. *Electronics Letters*, vol.16, N° 22, 1980, p. 860.

[179] Turenne (N.). – Apprentissage statistique pour l'extraction de concepts à partir de textres. application au filtrage d'informations textuelles. – Thèse de doctorat. ENSAIS, Université Louis-Pasteur Strasbourg, Novembre 2000.

[180] Turney (P. D.). – Mining the Web for Synonyms : PMI-IR versus LSA on TOEFL. *In : 12th European Conference on Machine Learning ECML 2001*. pp. 491–502. – Freiburg, Germany, 2001.

[181] Van Rijsbergen (C. J.). – *Information Retrieval, 2nd edition*. – Dept. of Computer Science, University of Glasgow, 1979.

[182] Vilalta (R.), Achari (M.) et Eick (C. F.). – Piece-Wise Model Fitting Using Local Data Patterns. *In : ECAI*, éd. par R. López de Mántaras and L. Saitta, IOS Press. Proceedings of the 16th European Conference on Artificial Intelligence, pp. 559–563. – Valencia, Spain, August 22-27 2004.

[183] Wang (W.), Yang (J.) et Muntz (R. R.). – STING : A Statistical Information Grid Approach to Spatial Data Mining. *In : Twenty-Third International Conference on Very Large Data Bases*, éd. par Jarke (M.), Carey (M. J.), Dittrich (K. R.), Lochovsky (F. H.), Loucopoulos (P.) et Jeusfeld (M. A.). pp. 186–195. – Athens, Greece, 1997.

[184] Wille (R.). – Restructuring Lattice Theory : an Approach Based on Hierarchies of Concepts. *In : Ordered Sets*, éd. par Rival (I.), pp. 445–470. – Dordrecht-Boston : Reidel, 1982.

[185] Wu (Z.) et Palmer (M.). – Verb semantics and lexical selection. *In : 32nd. Annual Meeting of the Association for Computational Linguistics*, pp. 133–138. – New Mexico State University, Las Cruces, New Mexico, 1994.

[186] Xie (X. L.) et Beni (G.). – A Validity measure for Fuzzy Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.23, N° 4, 1991.

[187] Xu (X.), Ester (M.), Kriegel (H. P.) et Sander (J.). – A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases. *In : 14th International Conference on Data Engineering*, pp. 324–331. – Orlando, FL, 1998.

[188] Yang (Y.) et Chute (C. G.). – An example-based mapping method for text categorization and retrieval. *ACM Trans. Inf. Syst.*, vol.12, N° 3, 1994, pp. 252–277.

[189] Yang (Yiming). – An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, vol.1, N° 1/2, 1997, pp. 69–90.

[190] Zahn (C.T.). – Graph theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, vol.20, 1971, pp. 68–86.

[191] Zargayouna (H.) et Salotti (S.). – SemIndex : a model of semantic indexing on XML documents. *In : 26th European Conference on Information Retrieval (ECIR'2004)*. – Sunderland, UK, 2004.

Une méthode de classification non-supervisée pour l'apprentissage de règles et la recherche d'information

Le regroupement d'objets, dans un cadre non-supervisé, est une tâche importante et difficile en apprentissage. Ce processus intervient dans des contextes variés tels que la découverte de connaissances, la simplification dans le représentation ou la description d'un ensemble de données.

Nous proposons, dans cette étude, l'algorithme de clustering PoBOC permettant de structurer un ensemble d'objets en classes non-disjointes. Nous utilisons cette méthode de clustering comme outil de traitement dans deux applications très différentes.

- En apprentissage supervisé, l'organisation préalable des instances apporte une connaissance utile pour la tâche d'induction de règles propositionnelles et logiques.
- En Recherche d'Information, les ambiguïtés et subtilités de la langue naturelle induisent naturellement des recouvrements entre thématiques.

Dans ces deux domaines de recherche, l'intérêt d'organiser les objets en classes non-disjointes est confirmé par les études expérimentales adaptées.

Mots clés : Apprentissage, classification, fouille de données, recherche d'information.

A Clustering method for rules learning and information retrieval

Data clustering is a major, but a hard, task in the unsupervised learning domain. This process is used in various context such as Knowledge Discovery, representation or description simplification of a data set.

In this study, we present the clustering algorithm PoBOC which organizes a dataset into overlapping classes which naturally match with real concepts of data. This clustering method is used in two very different applications.

- In the supervised learning field, the induction of a set of propositional and first-order rules is performed by first organizing each class into sub-classes.
- In the Information Retrieval field, the ambiguities from natural langage naturally induce overlaps between thematic.

On these two research domains, the organization of a dataset into overlapping clusters is validated with suitable experimental studies.

Keywords : Machine learning, classification, data mining, information retrieval.

Laboratoire d'Informatique Fondamentale d'Orléans - Université d'Orléans
rue Léonard de Vinci - BP 6759 - 45067 Orléans CEDEX 2