N° d'Ordre :
EDSPIC :

Université Paris 13
École Doctorale Galilée

THÈSE

présentée par

Guénaël CABANES

pour obtenir le grade de

Docteur d'université
Spécialité : Informatique

Classification non supervisée à deux niveaux guidée par le voisinage et la densité

Soutenue publiquement le 03/12/10 devant le jury :

Directeur de thèse :		
Y.	Bennani	Professeur, Université Paris 13
Rapporteurs :		
A.	Hardy	Professeur, Université de Namur
Y.	Lechevallier	Directeur de Recherche, INRIA
Examinateurs :		
M.	Aupetit	Chercheur, CEA
H.	Azzag	Maître de Conférences, Université Paris 13
G.	Cleuziou	Maître de Conférences, Université d'Orléans
A.	Cornuéjols	Professeur, AgroParisTech
J.L.	Deneubourg	Professeur, Université libre de Bruxelles
D.	Fresneau	Professeur, Université Paris 13

Résumé

Le travail de recherche exposé dans cette thèse concerne le développement d'approches à base de cartes auto-organisatrices (SOM) pour la découverte et le suivi de structures de classes dans les données par apprentissage non supervisé (clustering). Nous proposons une méthode de clustering à deux niveaux simultanés (DS2L-SOM). Cette méthode se base sur l'estimation, à partir des données, de valeurs de connectivité et de densité des prototypes de la SOM. Ces valeurs sont utilisées pour effectuer une classification des données. Le nombre de clusters est détecté automatiquement. De plus, la complexité est linéaire selon le nombre de données. Nous montrons aussi qu'il est relativement simple et efficace d'adapter DS2L-SOM aux variantes de l'algorithme SOM, de façon à obtenir une méthode très polyvalente capable par exemple d'analyser différents types de données. Nous proposons en outre une amélioration de la qualité de la SOM en utilisant les valeurs de connectivité lors de l'apprentissage des prototypes. Nous décrivons une nouvelle méthode de description condensée de la distribution des données, ainsi qu'une mesure heuristique de similarité entre ces modèles. Ces algorithmes se basent sur une estimation de la densité sous-jacente des données pendant l'apprentissage d'une SOM modifiée. La qualité de la description obtenue et de la mesure de comparaison est validée sur un ensemble de jeux de données artificiels et réels. Les propriétés de ces algorithmes rendent possible l'analyse de grandes bases de données, y compris de grands flux de données, qui nécessitent à la fois vitesse et économie de ressources. Par ailleurs, nous combinons l'algorithme de clustering à la mesure de similarité entre distributions pour l'analyse de données évolutives. Nous proposons un algorithme de suivi des données d'un flux permettant le stockage régulier de la structure des données, ainsi que la compression de ces informations au cours du temps. Les informations stockées peuvent ensuite être comparées entre elles pour l'analyse de l'évolution de la structure du flux de données. Enfin, nous présentons deux applications réelles pour le suivi d'individus dans un dispositif RFID. La première application est une étude biologique du comportement d'une colonie de fourmis pendant le déménagement d'un nid à l'autre. La deuxième application est une étude commerciale nécessitant le suivi de clients dans un grand magasin pendant leurs achats.

Abstract

The research outlined in this thesis concerns the development of approaches based on self-organizing maps (SOM) for the discovery and monitoring of class structures in the data by unsupervised learning (clustering). We propose a simultaneously two levels clustering method (DS2L-SOM). This method is based on the estimate from the data, values of connectivity and density of the prototypes of the SOM. These values are used to perform data classification. The number of clusters is detected automatically. Moreover, the complexity is linear with the number of data. We show that it is relatively simple and efficient to adapt to DS2L-SOM variants of the SOM algorithm in order to obtain a versatile method capable of analyzing such data types. We also propose an improvement of the quality of the SOM using the connectivity values during the learning of the prototypes. We describe a new method of condensed description of the data distribution, and a heuristic measure of similarity between these models. These algorithms are based on an estimate of the density underlying data for learning a modified SOM. The properties of these algorithms make possible the analysis of large databases, including large data flows that require both speed and economy of resources. In addition, we combine the clustering algorithm to measure similarity between distributions for the analysis of evolutionary data, and we propose an algorithm for monitoring data stream. Finally, we present two applications for tracking individuals in an RFID device. The first application is a biological study of the behavior of a colony of ants while moving from one nest to another. The second application is a business case requiring tracking of customers in a department store for their purchases. Finally, we propose an algorithm for monitoring data flow for storing regular data structure and compression of this information over time. The stored information can then be compared with each other to analyze the changing structure of the data stream.

Table des matières

Introduction **1**

1 Classification non supervisée **7**
1.1 Introduction 9
1.1.1 Caractéristiques des algorithmes de classification 9
1.2 Méthodes basées sur la distance 10
1.2.1 Algorithmes des K-Moyennes 11
1.2.2 Algorithmes de Classification Ascendante Hiérarchique 12
1.2.3 Cartes Auto-organisatrices et méthodes à deux niveaux 14
1.3 Méthodes basées sur les graphes 17
1.3.1 Créer un graphe à partir des données 18
1.3.2 CHAMELEON 19
1.3.3 Clustering Spectral 20
1.3.4 Propagation d'Affinité 22
1.3.5 NG + CHL : Méthode à deux niveaux basée sur les graphes . . 24
1.4 Méthodes basées sur la densité ou les probabilités 28
1.4.1 Mélange Gaussien 29
1.4.2 SOM : U*C 31
1.4.3 DBSCAN 32
1.5 Conclusion 34

2 Classification selon le voisinage et la densité **35**
2.1 Introduction 37
2.2 Classification selon le voisinage 37
2.2.1 Un nouvel algorithme de classification à deux niveaux : S2L-SOM 38
2.2.2 Validation 40
2.2.3 Conclusions 49

2.3 Classification selon la densité 49
2.3.1 Algorithmes 50
2.3.2 Validation 55
2.3.3 Comparaison avec des méthodes récentes 59
2.3.4 Conclusions 62
2.4 Conclusion 63

3 Relachement de contraintes topologiques et adaptation à d'autres types de données 67
3.1 Introduction 69
3.2 Relachement de contraintes topologiques 69
3.2.1 Notion de contraintes topologiques dans les SOM 71
3.2.2 Relâchement des contraintes topologiques guidé par les données 71
3.2.3 Résultats expérimentaux 74
3.2.4 Conclusion 82
3.3 Adaptation aux données intervalles 82
3.3.1 Les données intervalles 83
3.3.2 Adaptation de S2L-SOM aux données intervalles 85
3.3.3 Expériences et Résultats 86
3.4 Conclusion 91

4 Estimation et comparaison de distributions, application à la sélection de modèles 93
4.1 Introduction 95
4.2 Estimation de la densité des données 95
4.2.1 SOM - Enrichie 96
4.2.2 Estimation de la fonction de densité 99
4.2.3 Validation 101
4.3 Un outil de visualisation de la structure intra et inter groupes 103
4.3.1 Description de l'outil de visualisation utilisé 103
4.3.2 Applications 103
4.4 Une mesure pour la comparaison de distributions 108
4.4.1 Mesure de dissimilarité 109
4.4.2 Validation 110
4.4.3 Extention aux comparaisons de clusters 113
4.5 Sélection de modèle et Stabilité 116

4.5.1 La stabilité . . . 118
4.5.2 Approche proposée et protocole expérimental . . . 122
4.5.3 Résultats . . . 124
4.6 Conclusion . . . 127

5 Analyse de données évolutives 129
5.1 Introduction . . . 131
5.2 Analyse du comportement dynamique d'une colonie de fourmis . . . 131
5.2.1 Suivi par RFID d'une colonie de fourmis . . . 133
5.2.2 Structure des déplacement lors d'un déménagement . . . 137
5.3 Analyse des déplacements des clients dans un grand magasin . . . 150
5.3.1 Dispositif de suivi RFID . . . 150
5.3.2 Les données . . . 152
5.3.3 Pré-traitements des données . . . 152
5.3.4 Détection de la position du client et des changements de secteurs 153
5.3.5 Détection des patterns caractéristiques à l'échelle collective . . . 154
5.3.6 Résultats . . . 155
5.4 Analyse et classification des flux de données . . . 157
5.4.1 Protocole général de l'analyse des flux de données . . . 158
5.4.2 Un nouvel algorithme de clustering des flux de données . . . 159
5.4.3 Compression et stockage de la structure du flux . . . 165
5.5 Conclusion . . . 166

Conclusion 169

A Autres travaux réalisés 173
A.1 Bi-clustering . . . 173
A.1.1 Kdisj . . . 173
A.1.2 S2L-KDisj : Une extension de KDisj . . . 175
A.2 Version Batch généralisée . . . 180
A.2.1 Apprentissage des prototypes . . . 181
A.2.2 Enrichissement des prototypes . . . 182
A.2.3 Clustering automatique des prototypes . . . 182

Bibliographie 189

Table des figures

1.1 Exemple de clusters ne pouvant être correctement détectés par des algorithmes de type K-Moyennes. À gauche les deux clusters sont de taille très différentes, à droite il ne sont pas hypersphériques. 12
1.2 Arbre construit par Classification Hiérarchique. 13
1.3 Quelques étapes de l'apprentissage d'une carte auto-organisatrice. Les données sont dans la zone bleue, les prototypes sont en vert, reliés entre eux par des liens topologiques. A la fin de l'apprentissage les régions de Voronoï déterminent quel neurone sera le plus sensible pour chaque donnée. 16
1.4 Processus général d'une exécution de l'algorithme CHAMELEON. . . . 20
1.5 Exemple d'une exécution de l'algorithme Propagation d'Affinité [52]. Plus un point est rouge, plus il est candidat pour être un représentant. La couleur des arêtes est proportionnelle à la force des messages transmis. 25
1.6 Quelques étapes de l'apprentissage par Neural Gas + CHL. Les données sont dans la zone bleue, les prototypes sont en vert. Les connexions pertinentes sont créées au fur et à mesure. A la fin de l'apprentissage les régions de Voronoï déterminent quel prototype sera le plus représentatif de chaque donnée. 27
1.7 **Principe des Watersheds** : la matrice U^* est représentée sous la forme d'une surface topologique. Les zones de faible densité sont représentées en relief. Les bassins (zones à forte densité) sont virtuellement remplis d'eau. Les Watersheds sont les lignes de contacts entre l'eau issue des différents bassins, lorsque le niveau d'eau recouvre tout le relief. 32
1.8 **DBSCAN** : (a) p est atteignable à partir de q. (b) p et q appartiennent au même groupe 33

2.1 Données «Rings» et «Spirals» 41
2.2 Données «Chainlink» et «Atom» 42
2.3 Données «Diamonds» et «Hepta» 42

2.4 Données «Engytime» et «Wingnut» 42
2.5 Comparaison de la qualité de la segmentation avec l'indice de Jaccard selon les bases de données et les méthodes utilisées. 45
2.6 Comparaison de la stabilité de la classification selon les bases de données et les méthodes utilisées. 46
2.7 Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Hepta» 47
2.8 Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Spirals» 47
2.9 Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Random» 47
2.10 Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Chainlink» 48
2.11 Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Engytime» 48
2.12 Principe du calcul du seuil. À gauche : la densité minimale Dm entre les deux sous-groupes i et j de densité maximale D_i et D_j est inférieure au seuil $S(i, j)$, les deux sous-groupes ne sont pas fusionnés. À droite : la densité Dm est supérieure au seuil $S(i, j)$, les deux sous-groupes sont fusionnés. 53
2.13 Exemple de déroulement des différentes étapes de l'algorithme de raffinement 54
2.14 Valeur de la qualité de la segmentation selon l'indice de Jaccard pour chaque algorithme sur chaque base de données. 57
2.15 Valeur de l'indice de stabilité de la segmentation pour chaque algorithme sur chaque base de données. 58
2.16 Résultats obtenus à partir d'une CAH ou de DS2L-SOM sur les données "Ring". 59
2.17 Résultats obtenus à partir d'une CAH ou de DS2L-SOM sur les données "Engytime". 60
2.18 Résultats obtenus par (a) DBSCAN, (b) Analyse Spectrale, (c) CHAMELEON et (d) DS2L-SOM sur les données "TwoDiamonds". 61
2.19 Résultats obtenus par (a) Analyse Spectrale, (b) CHAMELEON et (c) DS2L-SOM sur les données "T4". 62
2.20 Résultats obtenus par (a) Analyse Spectrale, (b) CURE, (c) CHAMELEON et (d) DS2L-SOM sur les données "T7". 63
2.21 Résultats obtenus par (a) Analyse Spectrale, (b) CURE, (c) CHAMELEON et (d) DS2L-SOM sur les données "T8". 64

2.22 Visualisation (a) des prototypes de DS2L-SOM et (b) de la densité estimée sur les données "T8". 64

3.1 Dans FN-SOM des valeurs sont associées à chaque ligne et colonne d'une SOM de topologie rectangulaire. 70
3.2 Distance de Manhattan pondérée entre les neurones i et j pour une topologie hexagonale. Ici $d_M(i,j) = 3$, il s'agit du nombre minimal de connexions topologiques entre i et j. 72
3.3 Distance de Manhattan pondérée entre les neurones i et j pour une topologie hexagonale. Ici $d_{WM}(i,j) = 1.52 + 1.09 + 1.3 = 3.91$, il s'agit du plus court chemin entre i et j en fonction des valeurs de connexion v. 73
3.4 Visualisation des bases de données. 75
3.5 Visualisation de la valeur moyenne de Qe sur l'ensemble des bases de données pour différents relâchements des contraintes topologiques dans SOM. 77
3.6 Visualisation de la valeur moyenne de Ne sur l'ensemble des bases de données pour différents relâchements des contraintes topologiques dans SOM. 77
3.7 Visualisation de la valeur moyenne de Te sur l'ensemble des bases de données pour différents relâchements des contraintes topologiques dans SOM. 78
3.8 Visualisation de la valeur moyenne de Ge sur l'ensemble des bases de données pour différents relâchements des contraintes topologiques dans SOM. 78
3.9 Visualisation de la valeur moyenne de Qe sur l'ensemble des bases de données pour différentes valeurs de σ avec DDR-SOM. 79
3.10 Visualisation de la valeur moyenne de Ne sur l'ensemble des bases de données pour différentes valeurs de σ avec DDR-SOM. 80
3.11 Visualisation de la valeur moyenne de Te sur l'ensemble des bases de données pour différentes valeurs de σ avec DDR-SOM. 80
3.12 Visualisation de la valeur moyenne de Ge sur l'ensemble des bases de données pour différentes valeurs de σ avec DDR-SOM. 80
3.13 Visualisation des résultats obtenus avec SOM et DDR-SOM(1/2) pour deux bases de données. Les données sont en rouge et la grille de prototypes à la fin de l'apprentissage est représentée en noir. 81
3.14 Borne inférieure (a) et borne supérieure (b) d'un hyper-rectangle de dimension 3. 83

3.15 Exemple de mesure de distance L1 et L2. Ici $d_{L1}(x, x') = m_1 + m'_1$ et $d_{L2}(x, x') = \sqrt{{m_1}^2 + {m'_1}^2}$. 84
3.16 Exemple de mise-à-jour des prototypes. 85
3.17 Exemple d'initialisation "donnée". 87
3.18 Exemple d'initialisation "point". 87
3.19 Exemple d'initialisation "lininit". 87
3.20 Valeurs de l'indice de Jaccard pour différentes bases de données et différentes mesures de distances entre intervalles. 89
3.21 Valeurs de l'indice de Jaccard pour différentes bases de données et différentes initialisations de l'algorithme. 89
3.22 Visualisation des groupes obtenus pour les données "2Dim", "3Dim" et "5Dim" (seules les trois premières dimensions ont été représentées). . . 90
3.23 Visualisation des groupes obtenus pour les données "Soleil" et "Crochets". 90
3.24 Visualisation des groupes obtenus pour les données "Croix" et "Cible". 90

4.1 Données « Engytime » (gauche) et la fonction de densité estimée (droite). 102
4.2 Données « Wingnut » (gauche) et la fonction de densité estimée (droite). 102
4.3 Données « Rings » (gauche) et la fonction de densité estimée (droite). . 102
4.4 Données « Spirals » (gauche) et la fonction de densité estimée (droite). 102
4.5 Visualisation des données « Engytime ». 104
4.6 Visualisation des données « Hepta ». 104
4.7 Visualisation des données « Lsun ». 104
4.8 Visualisation des données « Rings ». 105
4.9 Visualisation des données « Spirales ». 105
4.10 Visualisation des données « Iris ». 106
4.11 Visualisation des données « Fourmis ». 106
4.12 Visualisation des données « Enfants ». 107
4.13 Exemple de comparaisons de la distribution des données. 108
4.14 Exemples de jeux de données pouvant être générés pour l'expérience. . 110
4.15 Visualisations de la matrice de dissimilarité obtenue entre différents jeux de données de distribution « Noise » 1 à 4. Les cellules sont d'autant plus sombre que la similarité est élevée. Les jeux de données sont triés selon leur distribution, ainsi les comparaisons de jeux de même distribution apparaissent selon quatre carrés sur la diagonale. Cette diagonale doit être la plus sombre possible pour une bonne performance de la mesure. 112

4.16 Visualisations des similarités entre modèles (un point = un modèle). Bleu : modèles de l'Anneau 5, Vert : Anneau 1, Turquoise : Anneau 3, Jaune et Orange : Spirales 1 et 2. 114
4.17 Dissimilarité entre les deux modèles de deux jeux consécutifs au cours du temps. Certains modèles ont été représentés pour illustrer les variations temporelles. Ces variations de la structure du flux (temps 5, 20 et 25) sont parfaitement détectées. 115
4.18 La classification est instable si le nombre de clusters est trop petit (ligne 1) ou trop grand (ligne 2). 118
4.19 Score de stabilité normalisé [156]. 121
4.20 Visualisations des valeurs moyennes de l'indice de Jaccard des résultats de DS2L-SOM sur données perturbées pour differentes valeurs du nombre de neurones. L'optimum est noté en rouge. 125

5.1 Dispositif RFID expérimental 134
5.2 Fourmi avec tag RFID 134
5.3 Classification par DS2L-SOM avec correction selon la densité à partir des données RFID. 135
5.4 Carte obtenue à partir des données RFID et profil des représentants. 136
5.5 Projection de Sammon des prototypes et leurs connexions, à partir des données RFID. 136
5.6 Le dispositif RFID expérimental et un exemple de détection enregistrée. 139
5.7 Exemple d'une séquence de déplacement. Le temps depuis le début de l'expérience (en minutes) est en abscisse. La position de la fourmi est en ordonnée. 140
5.8 Exemple de représentation d'un comportement spatial courant (seules 4 variables sont montrées ici). 141
5.9 Exemple d'une détection automatique de sous-séquences homogènes pour un individu. 142
5.10 Le HMM Structuré modélisant le comportement (c'est-à-dire les activités) des fourmis. (a) L'étiqueteur d'activité : un S-HMM à trois niveaux utilisé pour la modélisation du comportement de la colonie. (b) Un exemple de séquence étiquetée obtenue en utilisant l'étiqueteur. (c) Un modèle indépendant à deux niveaux a été appris pour chaque activité. (b) Un bloc de base codant le passage sous un capteur. (c) Un bloc de base modélisant la durée de stationnement dans une chambre ou un tunnel. 145

5.11 Evolution du nombre d'individus impliqués dans les différentes activités. 147
5.12 Panier avec une étiquette RFID. 151
5.13 Le dispositif RFID expérimental, les cases jaunes représentent la position des antennes dans le magasin. 151
5.14 Exemple d'enregistrement dans le fichier de données. 152
5.15 Exemple d'une séquence de déplacement (∼20 min) représentée par le numéro de l'antenne détectant le panier en fonction du temps (secondes). 153
5.16 Exemple de découpage automatique d'une trajectoire client. 154
5.17 Estimation de l'emplacement des différents secteurs. L'épaisseur des flèches est proportionnelle à la fréquence des transitions. Les fréquences les plus faibles n'ont pas été représentées. La taille de chaque secteur est proportionnelle au temps moyen passé à l'intérieur. 156

A.1 Disposition des salles dans la fourmilière. 178
A.2 Résultats de l'étude par ACM et tests statistiques. 179
A.3 Résultats de l'étude par S2L-KDisj. 180

Liste des tableaux

2.1 Description des données de test 41
2.2 Comparaison de la qualité de la segmentation avec l'indice de Jaccard moyen (arrondi au centième) selon les bases de données et les méthodes utilisées. . 45
2.3 Comparaison de la stabilité de la classification selon les bases de données et les méthodes utilisées. 46
2.4 Nombre de groupes obtenus avec différentes méthodes de classificassion (SL=SingleLinkage, KM=K-means, SSL=SOM+SingleLinkage, SKM=SOM+K-means). 56
2.5 Comparaison de la qualité de la segmentation avec l'indice de Jaccard moyen (arrondi au centième) selon les bases de données et les méthodes utilisées. . 56
2.6 Comparaison de la stabilité de la classification selon les bases de données et les méthodes utilisées. 58

3.1 Description des bases de données utilisées. 74
3.2 Valeurs de Ge obtenues par DDR-SOM pour chaque base de données avec différentes valeurs de σ. 79

4.1 Valeurs de l'indice de Dunn obtenues à partir de diverses mesures de dissimilarité pour comparer différentes distributions de données. 113
4.2 Valeurs de $Q(Nn_{opt})$, $Q(Nn_{jacc})$ et $Q(Nn_{heu})$ pour chaque base de données (en pourcentage). 126

A.1 Exemple de tableau disjonctif pour données qualitatives 174
A.2 Exemple de tableau en fréquence 176

Introduction

Contexte et problématique

La classification non supervisée est une approche importante en analyse exploratoire de données non étiquetées, mais reste un problème difficile [33]. Sans connaissances a priori sur la structure d'une base de données, seule la classification non supervisée permet de détecter automatiquement la présence de sous-groupes pertinents (ou *clusters*). Un cluster peut être défini comme un ensemble de données similaires entre elles et peu similaires avec les données appartenant à un autre cluster (homogénéité interne et séparation externe). Les clusters peuvent aussi être décrits comme des régions de l'espace de représentation des données contenant une densité relativement élevée de points de données, séparées entre elles par une zone de densité relativement faible. La détection de ces regroupements joue un rôle indispensable pour la compréhension de phénomènes variés décrits par un ensemble d'observations.

De nombreuses méthodes de classifications ont été proposées [77]. Les approches les plus classiques sont les méthodes hiérarchiques et les méthodes partitives. Bien que ces méthodes aient longtemps été très populaires, elles sont de plus en plus remplacées par des méthodes s'appuyant par exemple sur une carte auto-organisatrice ou Self Organizing Map (SOM) [86, 87].

Une SOM est un algorithme neuro-inspiré (inspiré du fonctionnement des neurones biologiques) qui permet la projection non linéaire de données de grandes dimensions dans un espace à deux dimensions par l'intermédiaire d'un apprentissage non supervisé compétitif. Cet algorithme est très efficace pour la réduction de dimensions et donc pour la visualisation des données sur une carte en deux dimensions [87, 14]. La carte est composée d'un ensemble de prototypes qui, à la fin de l'apprentissage, représentent les données et leur structure. On peut alors utiliser un algorithme de classification hiérarchique ou partitive pour effectuer une classification uniquement sur les prototypes [151], ce qui permet de segmenter la carte en zones représentatives des differents clusters. Cette approche est dite à deux niveaux, puisque dans un premier temps les données sont remplacées par un ensemble réduit de représentants (les prototypes de la carte), et dans un deuxième temps on effectue une classification de ces représentants.

Bien que les méthodes à deux niveaux soient plus intéressantes que les méthodes classiques (en particulier en réduisant le temps de calcul [151]), la classification obtenue à partir des représentants n'est pas optimale, puisqu'une partie de l'information a été perdue lors de la première étape. De plus, cette séparation en deux étapes n'est pas

adaptée à une classification dynamique de données qui évoluent dans le temps, malgré des besoins importants d'outils pour l'analyse de ce type de données.

Nous proposons dans cette thèse un ensemble de méthodes d'analyse à deux niveaux de la structure des données, basées sur l'apprentissage d'une SOM. L'objectif est d'obtenir un ensemble d'outils à la fois performants et rapides, capables de traiter différents types de données (vectorielles, évolutives, complexes, ...), pour des applications réelles (grandes bases de données, flux de données, ...).

Organisation de la thèse

Ce manuscrit est organisé en cinq chapitres principaux :

- **Chapitre 1 :** Ce chapitre est une présentation générale des principaux algorithmes pour la classification automatique des données. Après une description des différentes taxonomies utilisées pour regrouper les différents algorithmes, ceux-ci sont présentés selon la façon dont ils définissent une partition des données. Les principales limitations de ces algorithmes pour la découverte de clusters naturels sont soulignées dans ce chapitre, afin de motiver la proposition de nouveaux algorithmes. Parmi les algorithmes effectuant une partition des données selon la distance entre les clusters, nous proposons une description des méthodes à deux niveaux basées sur l'apprentissage d'une SOM. Les nouveaux algorithmes proposés dans ce manuscrit se basent tous sur l'apprentissage d'une SOM, qui est donc particulièrement détaillée dans ce chapitre. Nous présentons en outre des algorithmes populaires de classification se basant sur la connectivité ou sur la densité des données. Ces notions seront utilisées dans les nouveaux algorithmes proposés dans les chapitres suivants.
- **Chapitre 2 :** Ce chapitre décrit deux nouvelles méthodes de classification à deux niveaux simultanés adaptées aux données vectorielles. Les principaux avantages de ces deux algorithmes, testés sur un ensemble de jeux de données artificiels, sont d'une part leur vitesse d'exécution et leur capacité à détecter des clusters de forme arbitraire, et d'autre part leur capacité à déterminer automatiquement le nombre de clusters à détecter dans les données. Le premier algorithme estime une valeur de connectivité des prototypes de la SOM à partir des données pendant l'apprentissage de la carte. Ces valeurs estiment la force du voisinage entre deux prototypes. Elles permettent de construire un graphe reliant des ensembles de prototypes représentant le même type de données. Les composantes connexes du graphe ainsi

formé définissent les clusters. La principale limitation de cet algorithme est son inaptitude à discriminer des clusters en contact ou bruités. Nous proposons donc un deuxième algorithme, qui utilise une estimation de la densité des données pour détecter des frontières entre clusters définis par une zone de faible densité.

– **Chapitre 3 :** Dans ce chapitre sont exposées des extensions des algorithmes de classification proposés dans le Chapitre 2. La première extension vise à utiliser les valeurs associées aux connections entre prototypes pour améliorer la qualité de la représentation des données par la SOM, tout en conservant au maximum la topologie bidimensionnelle de la carte (nécessaire à la visualisation des résultats). La deuxième extension est une adaptation à des données non vectorielles : les données intervalles. Les modifications principales sont une nouvelle définition des prototypes et de la distance entre données et prototypes.
– **Chapitre 4 :** Ce chapitre présente une méthode à deux niveaux d'estimation de la distribution des données, ainsi qu'une mesure adaptée de comparaison de distributions et une application à la sélection de modèle selon le principe de stabilité. L'estimation de la distribution des données est obtenue par la construction d'une fonction de densité à partir d'une SOM enrichie d'informations de densité et de connectivité apprise à partir des données selon les principes mis en œuvre dans les chapitres précédents. La mesure de comparaison de distributions est une adaptation d'une mesure de divergence entre deux fonctions de densité. Les résultats obtenus à partir de tests sur des bases de données artificielles et réelles montrent la pertinence de l'approche. Le principal avantage de ces méthodes, outre leur compatibilité totale avec les algorithmes de classification proposées précédemment, est leur rapidité d'exécution et leur capacité à s'exécuter "en ligne" (une seule présentation des données suffit). Ces avantages permettent l'analyse de grandes bases de données ou de bases de données en évolution (tel que les flux de données).
– **Chapitre 5 :** Finalement, le dernier chapitre présente deux applications réelles de l'ensemble des algorithmes proposés précédemment pour l'analyse de données spatio-temporelles, ainsi qu'une proposition d'algorithme pour l'analyse de grand flux de données. La première application est une étude biologique visant à analyser la dynamique du comportement collectif d'une colonie de fourmis tropicales lors d'un déménagement (ANR Blanc, Sillages N° 05 BLAN 017701). La seconde application est une analyse du déplacement de clients dans un grand magasin pour la découverte des grandes zones de fréquentation et des fréquences de passages dans les différentes allées (ANR CADI N° 07 TLOG 003).

Nous concluons cette thèse en exposant les points forts de nos contributions et les perspectives de recherche dans ce domaine.

Définitions et notations

Dans de document de nombreux termes techniques seront fréquemment employés. Nous donnons ici une définition des notions les plus importantes, ainsi que le cas échéant les notations mathématiques utilisées dans ce manuscrit :

- **Données** : Une donnée x est la description élémentaire d'un phénomène à étudier. L'ensemble des données disponibles pour l'analyse du phénomène est appelé dans ce manuscrit "jeu de données" et est noté $X = \{x^{(1)}, \ldots, x^{(N)}\}$, N étant la taille du jeu de données.
- **Vecteurs** : La plupart des données utilisées dans ce manuscrit sont représentées par des vecteurs $x = (x_1, \ldots, x_d)$ de $\mathbb{R}^d$, d étant le nombre de variables numériques associées à chaque donnée. d est aussi appelé la "dimension" de l'espace de représentation des données. Lorsque les données sont des vecteurs, on parle parfois de "points de données".
- **Similarité** : La similarité entre deux éléments i et j est noté $s(i, j)$. La similarité entre les données à étudier est l'information principale (et souvent unique) permettant à l'algorithme de classification de partitionner le jeu de donnée. Le plus souvent, cette similarité est calculée selon une mesure de "distance" adaptée au type de données et au problème à résoudre. Une distance, noté de manière générale $d(i, j)$, est une mesure de similarité qui vérifie certaines propriétés mathématiques (symétrie, séparation et inégalité triangulaire). Lorsque les données sont décrites dans un espace vectoriel, la distance la plus utilisée est la distance Euclidienne, alors notée $\| i - j \|$.
- **Prototypes** : Un prototypes w est un représentant d'un ensemble de données. Un prototype est souvent du même type que les données (un vecteur par exemple), et il est toujours possible de définir une similarité entre un prototype et une donnée. En particulier, l'algorithme SOM utilisé dans cette thèse est basé sur l'apprentissage d'un ensemble de prototypes $W = \{w_1, \ldots, w_M\}$, M étant le nombre de prototypes. On parle ici d'apprentissage car les prototypes sont ajustés itérativement aux données pendant l'exécution de l'algorithme. À la fin de l'apprentissage des prototypes, on peut associer chaque donnée x à son prototype le plus représentatif (c'est à dire le plus similaire). On note ce prototype $u^*(x)$. Parfois, on définit aussi le deuxième prototype le plus représentatif d'une donnée, que l'on note alors $u^{**}(x)$.
- **Structure des données** : Lorsque nous parlons de "structure" des données dans cette thèse, nous faisons référence à la distribution sous-jacente du jeu de donnée. En particulier un algorithme de classification propose un *modèle* de cette distribution sous la forme d'une partition des données en clusters. Nous ne nous intéressons

ici qu'aux clusters "naturels" du jeu de données, c'est à dire aux clusters qui représentent des regroupements distincts dans la distribution des données.

Toute l'étude à été réalisée sous Matlab à partir d'une implémentation de l'algorithme SOM (SOM-ToolBox, [88]).

Chapitre 1

Classification non supervisée

1.1 Introduction

La classification non supervisée, ou apprentissage non supervisé, est un outil très important en analyse exploratoire de données non étiquetées. Il est utilisé pour la détection de regroupement, lorsque l'on n'a pas d'informations a priori sur la structure interne de ces données. Un problème de regroupement peut être défini comme le partitionnement d'un ensemble d'éléments en plusieurs sous groupes pertinents, en général mutuellement disjoints, les clusters. Les données regroupées dans un même cluster doivent être similaires entre elles (homogénéité interne), contrairement aux données appartenant à des groupes différents (séparation externe). Les méthodes de classification non supervisée jouent un rôle très important dans la compréhension de phénomènes variés décrits par des bases de données (voir par exemple [76, 159, 11, 30, 143]). Les applications les plus importantes sont la reconnaissance de la parole, la segmentation d'images, la fouille de textes, la catégorisation de clients, etc .... Des algorithmes de classification sont aussi beaucoup utilisés en géographie, en astronomie ou en génétique.

1.1.1 Caractéristiques des algorithmes de classification

Un grand nombre d'algorithmes on été proposés dans la littérature [77]. Ces différents algorithmes ont été regroupés selon différentes taxonomies suivant les caractéristiques prises en compte :

- **La représentation des données**. Chaque algorithme est en général adapté à un seul type de données. Les données peuvent être représentées sous différentes formes :
  - Vecteurs numériques ou catégoriques.
  - Séquences, arbres, graphes, etc ... (on parle souvent de données structurées).
  - Matrice de similarité.
- **Les méthodes de regroupements mises en œuvre** :
  - Méthodes *agglomératives*, les données sont itérativement ajoutées au cluster le plus pertinent.
  - Méthodes *divisives*, les données sont toutes regoupées en un seul cluster, puis l'algorithme divise itérativement ce cluster en clusters plus petits pour une meilleure représentation de la structure des données.
- **La forme de la partition obtenue** :
  - Méthodes *partitives*. L'algorithme essaye de produire une partition disjointe des données.

  - Méthodes *floues*. La partition n'est pas forcement disjointe, chaque donnée peut appartenir à plusieurs clusters.
  - Méthodes *hiérarchiques*. Les données sont regroupées hiérarchiquement sous la forme d'un arbre (ou dendrogramme), les nœuds de l'arbre issus d'un même parent forment des clusters.
- **Les critères de définition d'une partition**. Tous les algorithmes se basent sur une mesure de similarité entre données, dont le choix est très important pour la qualité des résultats obtenus. Il existe différentes manières d'utiliser cette similarité :
  - Partitionner selon la *distance* ou la similarité. Les clusters doivent être dissimilaires les uns des autres. La similarité entre deux clusters peut être définie de différentes façons. Le plus souvent on mesure la similarité entre les barycentres ou les médoïdes (élément le plus représentatif) des données appartenant à chaque cluster, ou entre les deux données les plus similaires appartenant chacune à un des clusters.
  - Partitionner selon la *connectivité*. Ces méthodes construisent un graphe à partir de la mesure de similarité entre données. Les clusters sont alors recherchés en minimisant la connectivité entre différents clusters et en maximisant cette connectivité au sein de chaque cluster, à l'aide d'outils d'analyse des graphes.
  - Partitionner selon la *densité*. Ces méthodes se basent sur une estimation de la densité des données dans l'espace de représentation en fonction de leur similarité. L'algorithme recherche des frontières entre clusters, ces frontières sont caractérisées par une densité faible par rapport à la densité de chaque cluster. Ainsi les zones de faible densité définissent les limites des clusters.

Dans la suite de ce chapitre nous allons présenter les principaux algorithmes de classification non supervisée fréquemment utilisés ou cités dans la littérature. Nous avons choisi de classer ces algorithmes selon leur critère de définition d'une partition.

1.2 Méthodes basées sur la distance

La plupart des algorithmes de clustering se basent sur une mesure de distance entre les données. Ces algorithmes minimisent en général une fonction de coût qui favorise la découverte de clusters compacts et bien séparés : les données d'un même groupe sont proches les unes des autres et sont éloignées des données des autres groupes.

Les approches les plus classiques sont les méthodes hiérarchiques et les méthodes partitives. Les méthodes de classification hiérarchiques agglomératives (CAH) utilisent

un arbre hiérarchique (dendrogramme) construit à partir des ensembles à classifier, les nœuds de l'arbre issus d'un même parent formant un groupe homogène [157]. Au contraire, les méthodes partitives (K-Moyennes par exemple) regroupent des données sans utiliser de structure hiérarchique. En général, dans ce cas, chaque cluster est représenté par un centre calculé dans l'espace des données.

1.2.1 Algorithmes des K-Moyennes

L'algorithme des K-Moyennes est probablement l'un des algorithmes de clustering les plus connus. Il est relativement simple et permet d'obtenir de bonnes performances grâce à la minimisation d'une fonction de coût $\widetilde{R}(c)$, avec certaines limitations qui seront discutées plus loin.

Algorithme des K-Moyennes [106] :

1) Déterminer le nombre K de clusters C_i et initialiser les centroïdes w_i aléatoirement (ces centres peuvent être choisis parmi les données).
2) Attribuer un numéro de cluster à chaque donnée selon le centre le plus proche.
3) Mettre à jour les centres des clusters de façon à minimiser la fonction de coût $\widetilde{R}(w)$:
$$\widetilde{R}(w) = \sum_{i=1}^{K} \sum_{x \in C_i} \| x - w_i \|^2$$
4) Tant que le partitionnement ne converge pas, retourner en 3.

De nombreuses adaptations des K-Moyennes ont été proposées [67, 80, 43, 46, 146], reposant toutes sur le calcul de centres représentatifs des différents clusters. Ce type d'algorithmes souffre de deux principaux inconvénients. Premièrement, le nombre de centres K doit être défini à priori. Cela nécessite soit de connaitre à l'avance le nombre de clusters que l'on souhaite obtenir, ce qui est rarement le cas si on veut découvrir des clusters naturels dans les données, soit de lancer l'algorithme un grand nombre de fois avec différentes valeurs de K et de choisir la meilleure segmentation selon un indice de qualité à définir (par exemple l'indice de Davies-Bouldin [35]), ce qui est particulièrement coûteux en temps de calcul et dépend de l'indice choisi. Deuxièmement, ces algorithmes ne peuvent pas découvrir des clusters contenant des données plus proches du centre d'un autre cluster que du centre de leur propre cluster. Cela arrive fréquemment dans les clusters naturels [81], par exemple s'il y a de grandes variations dans la

taille des clusters ou lorsque les clusters ne sont pas hypersphériques (voir la Figure 1.1 pour des exemples).

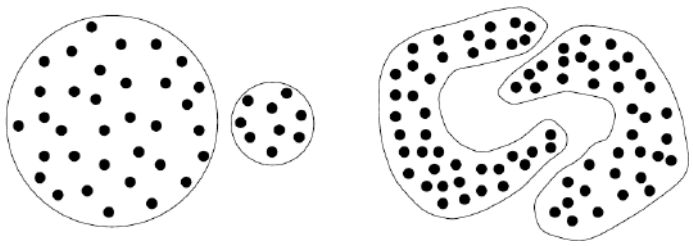


FIGURE 1.1 – Exemple de clusters ne pouvant être correctement détectés par des algorithmes de type K-Moyennes. À gauche les deux clusters sont de taille très différentes, à droite il ne sont pas hypersphériques.

1.2.2 Algorithmes de Classification Ascendante Hiérarchique

Les méthodes de Classification Ascendante Hiérarchique (CAH) utilisent un arbre hiérarchique (dendrogramme) construit à partir des ensembles à classifier. Les nœuds de l'arbre issus d'un même parent forment un groupe homogène [157, 142, 62, 78]. Il existe de nombreuses méthodes de création du dendrogramme, qui se basent toutes sur un algorithme général simple :

Algorithme de Classification Ascendante Hiérarchique :

1) Assigner à chaque donnée son propre numéro de cluster.
2) Calculer les similarités entre clusters selon une mesure préalablement choisie.
3) Fusionner les deux clusters les plus similaires et mettre à jour l'arbre hiérarchique (qui représente l'historique des fusions, Figure 1.2) .
4) Retourner en 2 jusqu'à ce que toutes les données soient regroupées en un unique cluster.
5) Sélectionner un niveau de coupure de l'arbre pour obtenir le nombre de clusters souhaité.

La principale source de variation entre les différents algorithmes proposés dans la littérature est le choix de la mesure de similarité entre deux clusters permettant de sélectionner les clusters à fusionner à chaque étape du processus. Les principales mesures proposées sont les suivantes :

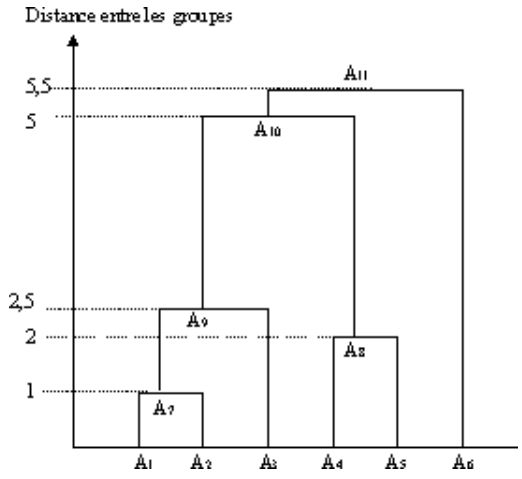


FIGURE 1.2 – Arbre construit par Classification Hiérarchique.

- *Le Lien Simple* : c'est une mesure de la distance minimale entre une donnée d'un cluster et une donnée de l'autre cluster. Cette mesure est très populaire. Elle permet en effet de détecter des clusters de forme quelconque, potentiellement non hypersphériques, ce qui est souvent nécessaire à la recherche de clusters naturels dans les données. En contrepartie cette mesure est très sensible au bruit et n'est pas capable de détecter des clusters en contact.
- *Le Lien Complet* : c'est une mesure de la distance maximale entre une donnée d'un cluster et une donnée de l'autre cluster. Cette mesure est très sensible au bruit et aux valeurs extrêmes. Elle est peu utilisée.
- *Le Lien Moyen* : c'est la distance moyenne entre une donnée d'un cluster et une donnée de l'autre cluster. Cette mesure est peu sensible au bruit mais a tendance à favoriser des clusters de formes hypersphériques.
- *La Distance aux Barycentres* : c'est la distance moyenne entre le barycentre des données d'un cluster et le barycentre de l'autre cluster. Cette mesure est très résistante au bruit et aux valeurs extrêmes, mais ne peut pas détecter des clusters de forme arbitraire. Une variante de cette mesure est la mesure de Ward [157] qui pondère la distance entre barycentres par le nombre de données faisant partie de chaque cluster. Cette mesure est une estimation de l'augmentation de la variance résultant d'une fusion des deux clusters.

L'algorithme CURE [62] est une adaptation de la Classification Ascendante Hiérarchique classique qui propose une mesure capable de détecter des clusters de forme arbitraire tout en étant résistant aux bruit et aux valeurs extrêmes. Pour cela, CURE calcule un nombre constant c de données représentatives de chaque cluster. Ces représentants sont calculés itérativement en sélectionnant la donnée la moins similaire au barycentre du cluster, puis en sélectionnant la donnée la moins similaire à celle

venant juste d'être choisie, et ainsi de suite jusqu'à l'obtention de c représentants. Les représentant d'un cluster sont "rapprochés" du barycentre d'un facteur λ constant. Une mesure de Lien Simple entre les représentants de deux clusters est finalement utilisée comme critère de fusion.

Il existe deux principales limitations à l'utilisation de ce type d'algorithmes pour l'analyse de données réelles. Premièrement, une fois le dendrogramme obtenu, il est nécessaire de choisir un niveau de coupure pour obtenir les clusters. Le choix de ce niveau de coupure reste un problème difficile malgré de nombreuses méthodes proposées (voir [78]). Deuxièmement, Tous ces algorithmes ont une complexité au minimum proportionnelle au carré du nombre de données, ce qui les rend inutilisables pour l'analyse de grandes bases de données.

1.2.3 Cartes Auto-organisatrices et méthodes à deux niveaux

Les algorithmes que nous proposons dans ce manuscrit sont tous basés sur l'apprentissage d'une carte auto-organisatrice ou Self Organizing Map (SOM, [86, 87]). Il s'agit d'un algorithme de quantification vectorielle qui est souvent utilisé comme première étape d'un processus de classification dit "à deux niveaux" comme cela est expliqué plus loin.

Une carte auto-organisatrice est un algorithme d'apprentissage compétitif non supervisé à partir d'un réseau de neurones artificiels. C'est une technique non linéaire très populaire pour la réduction de dimensions et la visualisation des données. Lorsqu'une observation est reconnue, l'activation d'un neurone du réseau, sélectionné par une compétition entre les neurones, a pour effet le renforcement de ce neurone et l'inhibition des autres (c'est la règle du "Winner Takes All"). Chaque neurone se spécialise donc au cours de l'apprentissage dans la reconnaissance d'un certain type d'observations. La carte auto-organisatrice est composée d'un ensemble de neurones connectés entre eux par des liens topologiques qui forment une grille bi-dimensionnelle. Chaque neurone est connecté à n entrées (correspondant aux n dimensions de l'espace de représentation) selon n pondérations $w_j = (w_{0j}, ..., w_{nj})$ (qui forment le vecteur prototype du neurone). Les neurones sont aussi connectés à leurs voisins par des liens topologiques. Le jeu de données est utilisé pour organiser la carte selon les contraintes topologiques de l'espace d'entrée. Ainsi, une configuration entre l'espace d'entrée et l'espace du réseau est construite ; deux observations proches dans l'espace d'entrée activent deux unités proches sur la carte. Une organisation spatiale optimale est déterminée par la SOM à partir des données et quand la dimension de l'espace d'entrée est inférieure à trois, aussi bien la position des vecteurs de poids que des relations de voisinage directes entre

les neurones peuvent être représentées visuellement. Pour chaque donnée présentée en cours d'apprentissage, le meilleur neurone (le plus sensible à cette donnée) met à jour son vecteur prototype w de façon à améliorer sa sensibilité à ce type de données. Pour assurer la conservation de la topologie de la carte, les autres neurones mettent à jour leurs prototypes de la même manière, mais selon une amplitude qui dépend de leurs distances par rapport au meilleur neurone. Ainsi, les prototypes les plus proches d'une donnée correspondent à des neurones voisins sur la carte.

L'apprentissage connexionniste est souvent présenté comme la minimisation d'une fonction de coût. Dans notre cas, cela correspond à la minimisation de la distance entre les données et les prototypes de la carte, pondérée par une fonction de voisinage K_{ij} [87]. Pour ce faire, nous utilisons un algorithme de gradient. La fonction de coût à minimiser est définie par :

$$\widetilde{R}(w) = \sum_{k=1}^{N} \sum_{j=1}^{M} K_{j,u^*(x^{(k)})} \parallel w_j - x^{(k)} \parallel^2$$

Avec N le nombre de données, M le nombre de neurones de la carte, $u^*(x^{(k)})$ est le neurone dont le vecteur prototype est le plus proche de la donnée $x^{(k)}$ (le "best match unit" : BMU). w_j est le prototype associé au neurone j. K_{ij} est une fonction symétrique positive à noyau : la fonction de voisinage. L'importance relative d'un neurone i comparé à un neurone j est pondérée par la valeur de K_{ij}, qui peut être définie ainsi :

$$K_{i,j} = \frac{1}{\lambda(t)} \times e^{-\frac{d_1^2(i,j)}{\lambda^2(t)}}$$

$\lambda(t)$ est une fonction de température qui contrôle l'étendue du voisinage qui diminue avec le temps t de λ_i à λ_f (par exemple $\lambda_i = 2$ à $\lambda_f = 0,5$) :

$$\lambda(t) = \lambda_i (\frac{\lambda_f}{\lambda_i})^{\frac{t}{t_{max}}}$$

t_{max} est le nombre maximum d'itérations autorisé pour l'apprentissage. $d_1(i,j)$ est la distance de Manhattan définie entre deux neurones i (de coordonnée (k,m)) et j (de coordonnée (r,s)) sur la grille de la carte :

$$d_1(i,j) = \parallel r - k \parallel + \parallel s - m \parallel$$

L'algorithme SOM est le suivant :

1) **Phase d'initialisation** :
   - Définir la topologie de la carte.
   - Initialiser aléatoirement tous les prototypes $w_j = (w_{0j}, ..., w_{nj})$ pour chaque neurone j.
2) **Phase de compétition** :
   - Présenter une donnée $x^{(k)}$ choisie aléatoirement.
   - Parmi les M neurones, choisir le meilleur, $u^*(x^{(k)})$, pour représenter cette donnée :

$$u^*(x^{(k)}) = \underset{1 \leq i \leq M}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

3) **Phase d'adaptation** :
   - Mettre à jour les prototypes w_j de chaque neurone j selon la règle :

$$w_j(t) = w_j(t-1) - \varepsilon(t) K_{ju^*(x^{(k)})} \left(w_j(t-1) - x^{(k)} \right)$$

   avec $\varepsilon(t)$ le taux d'apprentissage, qui diminue avec le temps.
4) **Répéter les phases 2 et 3** jusqu'à ce que les mises à jours des prototypes soient négligeables.

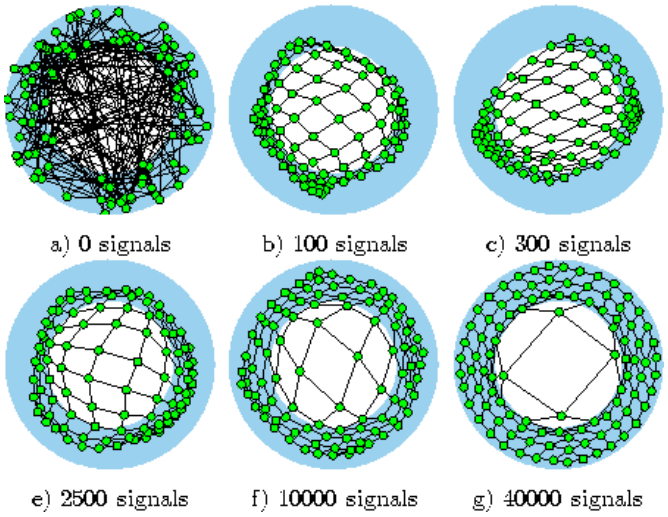


FIGURE 1.3 – Quelques étapes de l'apprentissage d'une carte auto-organisatrice. Les données sont dans la zone bleue, les prototypes sont en vert, reliés entre eux par des liens topologiques. A la fin de l'apprentissage les régions de Voronoï déterminent quel neurone sera le plus sensible pour chaque donnée.

Cet algorithme est souvent utilisée comme première étape d'une "classification à deux niveaux". Ce type de méthode procède en deux phases pour identifier les clusters dans un jeu de données. De nombreuses approches ont été proposées pour résoudre

des problèmes de classification à deux niveaux [19, 74, 149, 60, 89]. Dans la première phase du processus, l'algorithme estime des référents représentant des micro-groupes (micro-clusters). Dans la deuxième phase, les partitions associées à chaque référent sont utilisées pour former la classification finale des données en utilisant une méthode de classification traditionnelle. Ces approches sont particulièrement efficaces du fait de la vitesse d'apprentissage de SOM, de ses performances en réduction de dimensions non linéaire et la visualisation des résultats de la classification. A la fin de l'apprentissage, les prototypes de la carte représentent au mieux la structure des données. Lors de la deuxième étape (niveau 2), la détection des regroupements est en général obtenue en utilisant des techniques de classification classiques telles que K-Moyennes ou des méthodes hiérarchiques uniquement sur les prototypes [151], de façon à obtenir des regroupements de prototypes qui représentent la partition finale des données, puis on attribue à chaque donnée le numéro de cluster du prototype le plus proche de cette donnée. Cette approche est dite à deux niveaux, puisque dans un premier temps les données sont remplacées par un ensemble réduit de représentants (les prototypes de la carte), et dans un deuxième temps on effectue une classification de ces représentants.

L'application de ces algorithmes de classification sur un nombre réduit de prototypes est très intéressante en temps de calcul. De plus, la projection non linéaire en deux dimensions permet de visualiser efficacement la structure des données et les regroupements obtenus après classification [151].

Cependant, cette approche en deux étapes n'est pas optimale. En effet, au cours de l'apprentissage de la carte on perd une certaine quantité d'information, qui ne pourra pas être utilisée dans la phase de classification, en réduisant le nombre de dimensions à deux et en réduisant l'ensemble des données en un ensemble restreint de prototypes. De plus, cette séparation en deux étapes n'est pas adaptée à une classification dynamique de données qui évoluent dans le temps, malgré des besoins importants d'outils pour l'analyse de ce type de données.

1.3 Méthodes basées sur les graphes

Une autre approche proposée par de nombreux auteurs est de considérer les relations de distance ou de similarité entre données comme un graphe [104, 158, 62, 68]. Cette représentation est indépendante du type des données (vecteurs, textes, images, etc ...) puisque seule la similarité entre données est utilisée. Dans ce type de graphe, chaque nœud représente une donnée et chaque poids associé à un arc représente la distance ou

la similarité entre les deux données connectées. Un cluster est alors défini comme un ensemble de données fortement connectées entre elles (les valeurs associées aux arêtes sont grandes) et faiblement connectées aux autres données (faibles valeurs associées aux arêtes). Une telle représentation permet l'utilisation d'outils issus de la théorie des graphes. Nous présentons dans cette section trois algorithmes récents se basant sur une représentation sous forme de graphe et qui présentent de bonnes performances : CHAMELEON [81], le Clustering Spectral [137, 113] et la Propagation d'Affinité [52]. Nous finissons par la description d'une méthode de classification à deux niveaux basée sur la création d'un graphe entre prototypes : Neural Gas + Competitive Hebbian Learning [102, 101].

1.3.1 Créer un graphe à partir des données

Soit un ensemble de donnée $x^{(1)}, \ldots, x^{(N)}$ et une mesure de similarité $s_{ij} \geq 0$ entre chaque paire de points $x^{(i)}$ et $x^{(j)}$. Si on n'a pas d'autre information que la similarité entre les données, un *graphe de similarité* $G = (V, E)$ est une représentation intéressante des relations entre données. V est l'ensemble des sommets du graphe, chaque sommet représente une donnée. E est l'ensemble des arêtes de G reliant les données entre elles si leur similarité n'est pas nulle, chaque arête entre deux données $x^{(i)}$ et $x^{(j)}$ est pondérée par la valeur de s_{ij}.

Pour créer un tel graphe à partir des similarités s_{ij}, plusieurs méthodes ont été proposées (voir [155]) :

- **Le graphe complet** : Chaque donnée est connectée avec toutes les autres si la similarité n'est pas nulle. Tous les arcs sont pondérés par les valeurs de $S = (s_{ij})_{i,j=1,\ldots,N}$. En général cette méthode est utilisée seulement si la mesure de similarité traduit déjà le voisinage local entre données. On utilise souvent pour cela une fonction de similarité Gaussienne : $s(x^{(i)}, x^{(j)}) = exp(\frac{-\|x^{(i)}-x^{(j)}\|^2}{2\sigma^2})$, le paramètre σ contrôlant la taille du voisinage.
- **Le graphe ϵ-voisinage** : Dans ce cas, seules les données ayant une dissimilarité (ou une distance) inférieure à ϵ sont connectées dans le graphe. En général les arêtes du graphe ne sont pas pondérées, ce qui revient à créer un graphe complet sur une discrétisation binaire de la matrice de similarité.
- **Le graphe des k plus proches voisins** : dans ce type de graphe une donnée $x^{(i)}$ est connectée à une autre donnée $x^{(j)}$ si une des deux données fait partie des k plus proches voisins de l'autre. Contrairement à la méthode précédente, ce type de graphe représente correctement des clusters de densités différentes. Une variante de cette méthode est le graphe des k plus proches voisins mutuels, qui regroupe deux

données uniquement si chacune fait partie des k plus proches voisins de l'autre. Dans ce cas le graphe a tendance à connecter des données dans les régions de densité homogène et de ne pas connecter des régions de densités différentes.

1.3.2 CHAMELEON

CHAMELEON [81] est une méthode Ascendante Hiérarchique basée sur une représentation sous forme de graphe des k plus proches voisins. La particularité de l'algorithme est de modéliser à la fois l'*inter-connectivité relative RI* et la *proximité relative RC* entre deux sous-graphes pour décider de leur fusion au cours du processus de clustering. RI est globalement une mesure de la force totale des connexions entre les sous-graphes alors que RC est une mesure de la force moyenne de ces connexions.

L'*inter-connectivité relative RI* entre deux sous-graphes est une fonction de la somme des poids des arêtes $EC(C_i, C_j)$ reliant les deux sous-graphes C_i et C_j. RI est *relative* à chaque paire de groupes car elle dépend aussi d'une mesure de l'*inter-connectivité interne* $EC(C_i)$ de chaque sous-graphe C_i. Cette mesure est une estimation de la somme des poids des arêtes reliant en moyenne deux sous-ensembles dichotomiques de C_i. Cette pondération permet à la mesure d'être adaptée aux différents degrés de connectivité pouvant caractériser différents clusters.

$$RI(C_i, C_j) = \frac{2EC(C_i, C_j}{EC(C_i) + EC(C_j)}$$

La *proximité relative RC* entre deux sous-graphes est une fonction de la moyenne des poids des arêtes $SEC(C_i, C_j)$ reliant les deux sous-graphes C_i et C_j. Sur un graphe des k plus proches voisins, SEC est une bonne mesure de l'affinité entre données situées à la frontière entre les deux ensembles. Elle est aussi résistante au bruit. SEC est pondérée par le nombre de sommets $|C_i|$ et $|C_j|$ de chacun des deux sous-graphes et par une mesure de *proximité interne* $SEC(C_i)$ calculée comme pour EC (en prenant la moyenne au lieu de la somme).

$$RC(C_i, C_j) = \frac{SEC(C_i, C_j)}{\frac{|C_i|}{|C_i|+|C_j|} SEC(C_i) + \frac{|C_j|}{|C_i|+|C_j|} SEC(C_j)}$$

L'algorithme CHAMELEON est le suivant :

Entrée : Une matrice de similarité $S = (s_{ij})_{i,j=1,\dots,n}$ avec n le nombre de données.

Sortie : Une segmentation hiérarchique des données à partir de petits clusters non hiérarchiques.

1) Construire le graphe des k plus proches voisins à partir de S, k étant un paramètre à choisir.
2) Segmenter le graphe avec l'algorithme hMetis [82] pour obtenir de petits sous-graphes bien connectés.
3) Construire une Classification Ascendante Hiérarchique de ces petits sous-graphes en fusionnant itérativement les paires C_i et C_j maximisant, pour un paramètre α constant, $RI(C_i, C_j) \times RC(C_i, C_j)^\alpha$.

L'arbre Hiérarchique obtenu peut être découpé selon différents critères pour obtenir une segmentation des données. La figure 1.4 donne un schéma général du processus.

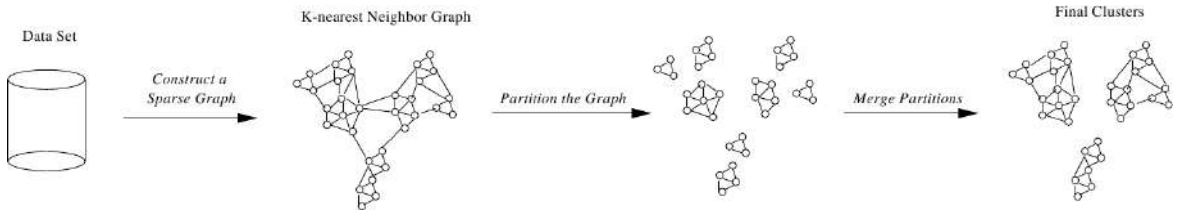


FIGURE 1.4 – Processus général d'une exécution de l'algorithme CHAMELEON.

Contrairement à la plupart des autres algorithmes Hiérarchiques, CHAMELEON est capable de s'adapter à différentes formes et densités dans les clusters, ou à la présence de bruit, ce qui en fait une méthode potentiellement très robuste pour l'analyse de données réelles. Cependant, cet algorithme souffre d'une complexité quadratique lorsque le nombre de données est important (la complexité est moindre pour de petites bases de données). Il n'est donc pas adapté à l'analyse de bases de données de grande taille. De plus, le choix de la coupure de l'arbre reste un problème difficile à résoudre, comme pour toutes les méthodes Hiérarchiques (voir la Chapitre 1.2). Il faut aussi noter que cet algorithme n'a été testé que sur des bases de données artificielles en deux dimensions. Ses performances sur des bases de données réelles de grandes dimensions sont inconnues.

1.3.3 Clustering Spectral

La matrice de similarité entre données $S = (s_{ij})_{i,j=1,\dots,N}$ est aussi la matrice des poids des arêtes, appelé *matrice d'adjacence* de G. Le *degré* d'un nœud de G est défini

tel que :

$$d_i = \sum_{j=1}^{N} w_{ij}$$

La *matrice des degrés D* est alors la matrice diagonale ayant les degrés de $d_1, \ldots, d_N$ sur la diagonale.

Le principal outil du Clustering Spectral est le Laplacien d'un Graphe, qui peut être défini de la façon suivante :

$$L = D - W$$

Il est souvent préférable d'utiliser un Laplacien normalisé [137, 113], il en existe deux types :

$$\begin{aligned} L_{sym} &= D^{-1/2} L D^{-1/2} \\ L_{rw} &= D^{-1} L \end{aligned}$$

Une propriété très intéressante du Laplacien est que le nombre de valeurs propres nulle de L est égal au nombre de sous-ensembles du graphe non connecté aux autres sommets. Chaque sous-ensemble étant un cluster bien séparé de l'ensemble des données, aussi appelé composante connexe pour un graphe. De plus, tout vecteur propre de L associé à une valeur propre nulle est un vecteur de taille N (le nombre de données) prenant une valeur constante non nulle pour chaque donnée appartenant a une des composantes connexes et une valeur nulle pour les autres données. Chacun de ces vecteurs propres est ainsi une représentation d'une composante connexe, l'ensemble des vecteurs propres associés à une valeur propre nulle représente l'ensemble des composantes connexes du graphe.

De la même façon, si le graphe est divisé en k sous-ensembles faiblement connectés, les k plus petites valeurs propres de L correspondront à des vecteurs propres ayant des valeurs plus fortes pour les sommets d'un des sous-ensembles que pour les autres sommets. Ces vecteurs sont appelés les k premiers vecteurs propres de L. L'idée de l'analyse spectrale est donc de choisir le nombre k de groupes souhaités puis de représenter chaque donnée dans un espace à k dimensions où les coordonnées correspondent aux valeurs des k premiers vecteurs propres. Dans cet espace, les données fortement connectées entre elles seront très proches (voire similaire si l'ensemble est une composante connexe) et les ensembles faiblement connectés seront très éloignés. Un simple k-moyenne suffit alors à discriminer les différents clusters.

L'algorithme est le suivant :

Entrée : Une matrice de similarité S et un nombre k de clusters.

Sortie : Une segmentation des données

1) Construire le graphe à partir de S.
2) Calculer le Laplacien L du graphe.
3) Calculer les k premiers vecteurs propres $v_1, \ldots, v_k$ de L.
4) Soit $M \in \mathbb{R}^{N \times k}$ la matrice contenant les vecteurs $v_1, \ldots, v_k$ en colonne. Pour tout $i = 1, \ldots, N$, définir un vecteur $y_i \in \mathbb{R}^k$ correspondant à la $i^{\text{ème}}$ ligne de M.
5) Segmenter les points $(y_i)_{i=1,\ldots,N}$ dans $\mathbb{R}^k$ avec k-moyennes pour obtenir une segmentation des données.

Si le Laplacien L_{sym} est utilisé, il est nécessaire de normaliser M tel que la somme de chaque ligne soit égale à 1 (voir [113]).

Le principal avantage de l'analyse spectrale est qu'il est possible de détecter des clusters de formes arbitraires. En effet de tels clusters sont représentés dans le graphe comme des composantes fortement connectées. Ces composantes seront projetées par l'algorithme sous une forme hypersphérique facilement détectée par un algorithme comme K-Moyennes. Il existe deux inconvénients majeurs à cette méthode. Premièrement on peut noter une grande sensibilité au bruit et plus généralement une difficulté à détecter des clusters en contact. Deuxièmement la complexité des calculs mis en jeux ne permet pas d'utiliser une analyse spectrale pour de grandes bases de données. En effet, l'étape principale du processus est le calcul des vecteurs et valeurs propres d'une matrice de taille $N \times N$, N étant le nombre de données. Sur de grandes matrices, cette étape peut être extrêmement couteuse en temps de calcul [155].

1.3.4 Propagation d'Affinité

L'algorithme de Propagation d'Affinité est un autre algorithme très populaire se basant sur une représentation sous forme de graphe [52]. L'idée principale est de partir d'un graphe (souvent complet) des données et de faire transmettre des messages entre les données le long des arrêtes en fonction de la valeur de ces arrêtes (c'est à dire la similarité s entre les données, ici en valeurs négatives). Ces échanges doivent permettre de déterminer quelles données sont de bons représentants locaux et quels représentants modélisent le mieux chacune des autres données. Les données s'échangent deux types

de messages le long du graphe. La "responsabilité" $r(i,k)$ envoyé de i vers k représente la qualité de k comme représentant de i par rapport aux autres représentants disponibles. La "disponibilité" $a(i,k)$ envoyé de k vers i représente la qualité de k comme représentant de i par rapport à l'existence d'autres données bien représentées par k et l'absence de bon représentant de k. La qualité totale de la représentation de i par k est donnée par $r(i,k) + a(i,k)$.

Les valeurs de chacun des deux types de messages dépendent des valeurs de l'autre type. Ainsi $r(i,k)$ dépend de la similarité entre k et i, ainsi que de la similarité entre i et d'autres représentants potentiels k' selon leur disponibilité $a(i,k')$:

$$r(i,k) = s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\}$$

De même, $a(i,k)$ dépend de l'auto-responsabilité de k, $r(k,k)$, qui est d'autant plus élevée que k ne possède pas de bons représentants. Elle dépend aussi de la qualité de la représentation de k pour d'autres données i', $r(i',k)$:

$$a(i,k) = \min\{0, r(k,k) + \sum_{i' \notin \{i,k\}} \max\{0, r(i',k)\}$$

Seule la portion positive de $r(i',k)$ est conservée de façon à ce que l'existence de données faiblement représentées par k ne pénalise pas le fait k représente fortement un certain nombre de données locales. $a(i,k)$ reste toujours négative pour limiter l'influence de fortes responsabilités positives.

L'algorithme de Propagation d'Affinité est le suivant :

Entrée : Une matrice de similarité $S = (s_{ij})_{i,j=1,\ldots,N}$ avec N le nombre de données et une valeur $s(k,k)$ associée à chaque donnée k.

Sortie : Une segmentation des données et une liste des représentants des cluster.

1) Construire le graphe à partir de S.
2) Pour toute paire de points (i,k) voisins sur le graphe, initialiser les disponibilité $a(i,k)$ à 0 et initialiser les responsabilités $r(i,k)$ telles que :

$$r(i,k) = s(i,k) - \max_{k' \neq k}\{s(i,k')\}$$

3) Jusqu'à convergence, faire :

3.1) Mettre à jours les disponibilités selon un facteur d'amortissement $\lambda \in [0, 1]$:

$$\begin{aligned} a_t(i,k) &= \lambda.a_{t-1}(i,k) + (1\text{-}\lambda).\left[\min\{0, r(k,k) + \sum_{i' \notin \{i,k\}} \max\{0, r(i',k)\}\right] \\ a_t(k,k) &= \lambda.a_{t-1}(k,k) + (1\text{-}\lambda).\left[\sum_{i' \notin \{i,k\}} \max\{0, r(i',k)\}\right] \end{aligned}$$

3.2) Mettre à jours les responsabilités selon λ :

$$\begin{aligned} r(i,k) &= \lambda.r_{t-1}(i,k) + (1\text{-}\lambda).\left[s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\}\right] \\ r(k,k) &= \lambda.r_{t-1}(k,k) + (1\text{-}\lambda).\left[s(k,k) - \max_{k' \neq k}\{a(k,k') + s(k,k')\}\right] \end{aligned}$$

4) Pour chaque donnée i, trouver le meilleur représentant k, celui qui maximise $a(i,k) + r(i,k)$. Si une donnée k^* est son propre meilleur représentant, elle défini un cluster.

5) Associer chaque donnée i à un représentant définissant un cluster en maximisant $a(i,k^*) + r(i,k^*)$, de façon à obtenir une segmentation des données.

Les valeurs $s(k,k)$ correspondent à une information à priori de la potentialité de chaque donnée à être un bon représentant local. Ces valeurs influencent le nombre total de clusters obtenus, plus elles sont globalement petites, plus le nombre de clusters sera petit, et inversement.

Le principal avantage de la Propagation d'Affinité est qu'il n'est pas nécessaire de définir à priori le nombre de clusters que l'on souhaite obtenir. Ce paramètre est obtenu automatiquement pendant le processus en fonction des valeurs $s(k,k)$ initialement associées aux données. Cependant le principal inconvénient de cet algorithme est sa lenteur. En effet le nombre de messages transmis est proportionnel au nombre d'arêtes du graphe, soit au pire une complexité en N^2. Une telle complexité limite l'analyse à des ensembles de données de petites tailles et n'est pas adaptée aux grandes bases de données.

1.3.5 NG + CHL : Méthode à deux niveaux basée sur les graphes

Certaines méthodes à deux niveaux se basent sur la création d'un graphe à partir d'un ensemble de prototypes représentatifs des données, c'est le cas de l'algorithme

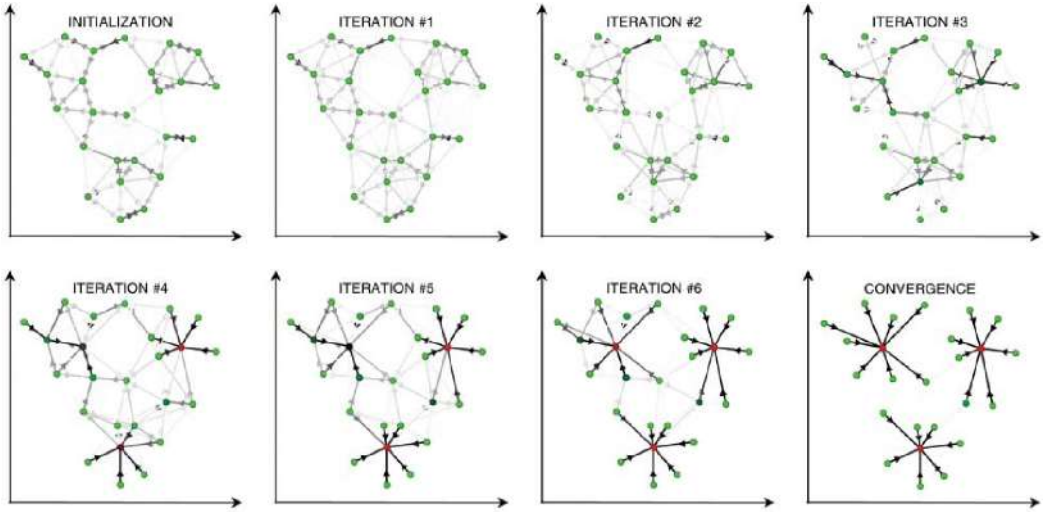


FIGURE 1.5 – Exemple d'une exécution de l'algorithme Propagation d'Affinité [52]. Plus un point est rouge, plus il est candidat pour être un représentant. La couleur des arêtes est proportionnelle à la force des messages transmis.

"Neural Gas + Competitive Hebbian Learning" (NG +CHL [102, 101]). Le premier niveau (représentation et compression des données) est assuré par l'algorithme NG [102] qui calcule un nombre important de prototypes à partir des données :

1) **Phase d'initialisation** :
   - Initialiser aléatoirement tous les prototypes $w_j = (w_{0j}, ..., w_{nj})$.
2) **Phase de compétition** :
   - Présenter une donnée $x^{(k)}$ choisie aléatoirement.
   - Parmi les M prototypes, choisir le meilleur $u^*(x^{(k)})$:

$$u^*(x^{(k)}) = \underset{1 \leq i \leq M}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

3) **Phase d'adaptation** :
   - Pour chaque prototype i déterminer k_i, le nombre de prototypes j tel que :

$$\parallel w_j - x^{(k)} \parallel^2 < \parallel w_i - x^{(k)} \parallel^2$$

   - Mettre à jour les prototypes w_j selon la règle :

$$w_j(t) = w_j(t-1) - \varepsilon(t) e^{-\frac{k_i}{\lambda(t)}} (w_j(t-1) - x^{(k)})$$

4) **Répéter les phases 2 et 3** jusqu'à ce que les mises à jours des prototypes soient négligeables.

Le deuxième niveau se fait pendant l'apprentissage des prototypes, ce qui minimise la perte d'information lors de leur classification. L'idée principale est simple : il s'agit de créer des connexions entre les prototypes qui représentent le même type de données et/ou de détruire les connexions reliant des prototypes représentant des données non similaires.

Cette étape est effectuée par l'algorithme CHL [101]. Elle consiste à déterminer pour chaque donnée les deux prototypes les plus représentatifs et de les connecter. Chaque connexion est associée à un âge qui augmente avec le temps. A chaque fois que deux neurones sont les plus sensibles pour une donnée, l'âge de leur connexion est mise à zéro. Quand une connexion est trop âgée elle est détruite. Ainsi, à la fin de l'apprentissage, chaque groupe de neurones connectés est un représentant d'un type de données similaires.

L'algorithme Neural Gas + CHL est donc le suivant :

1) **Phase d'initialisation** :
   - Initialiser aléatoirement tous les prototypes $w_j = (w_{0j}, ..., w_{nj})$.
   - Initialiser les connexions ν entre chaque couple de prototypes i et j :

$$\forall i, j \;\; \nu_{ij} = 0$$

2) **Phase de compétition** :

   - Présenter une donnée $x^{(k)}$ choisie aléatoirement.
   - Parmi les M prototypes, choisir les deux meilleurs $u^*(x^{(k)})$ et $u^{**}(x^{(k)})$ pour représenter cette donnée :

$$u^*(x^{(k)}) = \underset{1 \leq i \leq M}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

$$u^{**}(x^{(k)}) = \underset{1 \leq i \leq M,\ i \neq u^*}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

   - Créer une connexion si besoin entre $u^*(x^{(k)})$ et $u^{**}(x^{(k)})$, mettre l'âge A de cette connexion à zéro :

$$\nu_{u^* u^{**}} = 1 \ , \ A_{u^* u^{**}} = 0$$

   - Augmenter l'âge de toutes les connexions issues de $u^*(x^{(k)})$ et détruire celles ayant dépassé un âge limite T :

$$\forall j \ tel \ que \ \nu_{u^* j} = 1 \ , \ A_{u^* j} = A_{u^* j} + 1$$

$$\forall j \; tel \; que \; A_{u^*j} > T \; , \; \nu_{u^*j} = 0$$

3) **Phase d'adaptation** :

   – Pour chaque prototype i déterminer k_i, le nombre de prototypes j tel que :

$$\| w_j - x^{(k)} \|^2 < \| w_i - x^{(k)} \|^2$$

   – Mettre à jour les prototypes w_j selon la règle :

$$w_j(t) = w_j(t-1) - \varepsilon(t) e^{-\frac{k_i}{\lambda(t)}} (w_j(t-1) - x^{(k)})$$

4) **Répéter les phases 2 et 3** jusqu'à ce que les mises à jours des prototypes soient négligeables.

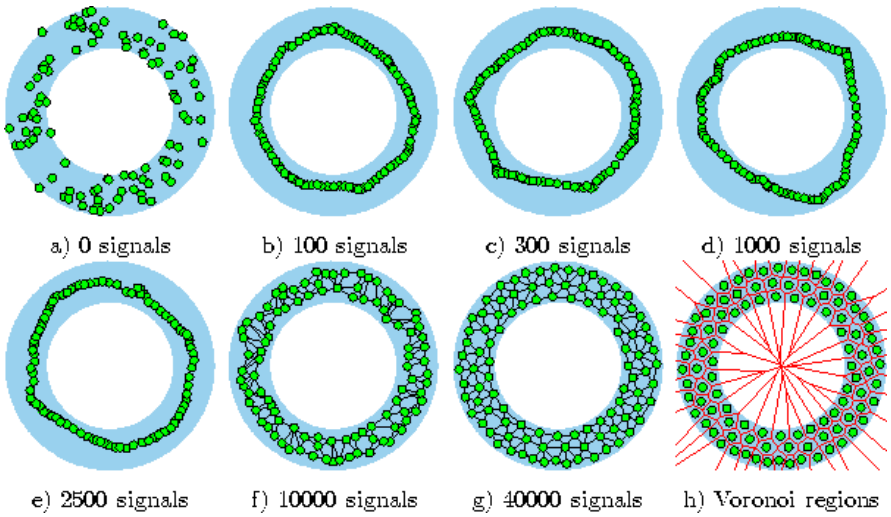


FIGURE 1.6 – Quelques étapes de l'apprentissage par Neural Gas + CHL. Les données sont dans la zone bleue, les prototypes sont en vert. Les connexions pertinentes sont créées au fur et à mesure. A la fin de l'apprentissage les régions de Voronoï déterminent quel prototype sera le plus représentatif de chaque donnée.

Cette méthode a de nombreux avantages. En particulier elle découvre les clusters en cours d'apprentissage et à partir des données, sans à priori sur leur répartition ou leur

structure (comme par exemple le nombre de clusters attendu). C'est aussi une méthode simple avec de bonnes performances en temps de calcul.

Par contre les résultats sont très sensibles à la présence de bruit dans les données, qui a tendance à favoriser la création de connexions non pertinentes. De plus, par rapport à des méthodes basées sur l'apprentissage d'une SOM, NG + CHL n'impose aucune contrainte sur la topologie des prototypes, ce qui ne permet pas de réduire le nombre de dimensions et de visualiser simplement la structure des données. Enfin, on n'a aucune information sur la structure générale des clusters entre eux, on ne sait pas par exemple si le cluster 1 est plus similaire au cluster 2 que le cluster 3.

En résumé, les algorithmes basés sur la représentation sous forme de graphes sont reconnus pour les bonnes performances du clustering obtenu. En particulier les clusters peuvent être reconnus quels que soient leurs formes, contrairement à la plupart des méthodes basées sur le calcul de centres représentatifs des clusters (K-Moyennes par exemple). Cependant, le problème majeur avec ce type d'algorithmes est la complexité des calculs mis en œuvre. En effet, le nombre d'arêtes à considérer est souvent proportionnel au carré du nombre de données, ce qui peut poser problème dans le cas de grandes bases de données. Les méthodes à deux niveaux ont cependant l'avantage de réduire considérablement cette complexité, puisque le nombre d'arêtes est alors proportionnel au nombre de prototypes, qui est en général bien plus petit que le nombre de données.

1.4 Méthodes basées sur la densité ou les probabilités

Une autre famille de méthodes de clustering se base non plus sur les distances entre données, mais sur les fluctuations de densité de l'espace de représentation des données. Il existe de nombreuses méthodes de classification à partir de la densité [44, 161, 149, 9, 115, 116]. L'idée générale est que les groupes à découvrir sont constitués d'un ensemble de points de forte densité qui forme le centre du groupe, entourés par des points en densité plus faible qui forment la périphérie. Dans cette optique, les zones de faibles densités locales (par rapport aux zones voisines) définissent les limites entre les groupes.

1.4.1 Mélange Gaussien

La densité de probabilité est un concept clé en statistique. Pour une base de donnée, la fonction de probabilité f donne une description naturelle de la distribution de ces données. L'estimation de la densité est alors la construction d'une estimation de la fonction de densité à partir de l'ensemble de donnée.

Une des méthodes les plus populaires pour estimer une fonction de densité est l'estimateur à Noyaux. Un estimateur de noyau K est défini par :

$$\hat{\mathrm{f}}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x^{(i)}}{h}\right)$$

Ainsi, l'estimateur à base de noyaux est un mélange de N noyaux, centrés sur les N données. h est le paramètre de lissage. Lorsque h tend vers 0, le nombre de modes de la fonction de densité tend vers N. Si on augmente h le nombre de modes tend vers un [138]. Le choix de la valeur de ce paramètre est la tâche la plus difficile dans l'estimation de la densité [127, 22, 128]. La complexité du choix optimal est en $O(N^2)$ [136], ce qui peut poser problème pour l'analyse de grandes bases de données.

Le noyau le plus largement utilisé est le Noyaux Gaussien.

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$$

Pour ce noyau, l'estimation de la densité s'écrit :

$$\hat{\mathrm{f}}(x) = \frac{1}{N\sqrt{2\pi}h} \sum_{i=1}^{N} e^{-\frac{(x-x^{(i)})^2}{2h^2}}$$

L'estimation par noyaux Gaussien peut être étendu aux dimensions multiples ($x \in \Re^d$) :

$$\hat{\mathrm{f}}(x) = \frac{1}{N(2\pi)^{d/2}h^d} \sum_{i=1}^{N} e^{-\frac{1}{2}\left(\frac{x-x^{(i)}}{h}\right)' \Sigma^{-1} \left(\frac{x-x^{(i)}}{h}\right)}$$

Avec Σ une matrice symétrique, positive-définie et $|\Sigma| = 1$, qui représente la forme (ou "l'orientation") du noyau [133]. Une estimation de h peut être obtenue selon une

heuristique [132, 133] :

$$h = N^{-1/(d+4)}$$

Une cross-validation complète peut aussi être utilisée pour estimer h à partir des données, mais cette méthode est coûteuse en temps de calcul. L'estimation d'un Σ optimal est généralement impossible, du fait d'un trop grand nombre de paramétres à optimiser en grandes dimensions [128]. Σ est donc souvent réduite à la matrice identité (Noyau Gaussien sphérique).

Une alternative à l'approche précédente est le mélange de modèles, dans lequel la densité sous-jacente est modélisée sous la forme :

$$\hat{\mathrm{F}}(x) = \sum_k \pi_k \varphi_k(x; \theta_k)$$

Les π_k sont les poids des noyaux. Les composants du mélange, φ_k, sont eux même des fonctions de densité de paramètres θ_k. Souvent, les φ_k sont choisis parmis des fonctions normales sphériques multivariées, dans ce cas $\theta_k = \{\mu_k, h_k\}$. Les mélanges de noyaux sont souvent motivés par une hétérogénéité ou la présence de populations distinctes dans les données [99, 105, 48]. La flexibilité du modèle en fait un candidat de choix pour l'estimation générale de la densité et l'analyse exploratoire. Le nombre de composants joue aussi comme un paramètre de lissage. Une plus grande flexibilité peu être obtenue en augmentant le nombre de composants, mais trop de composants peuvent mener à un sur-apprentissage et des variation excessives. Un modèle paramétrique est à un bout de ce spectre, un estimateur à Noyaux est à l'autre bout. Par exemple, un estimateur à Noyaux Gaussien peut être considéré comme un mélange de N Noyaux Gaussiens de poids $1/N$ et centrés sur chaque donnée.

La méthode la plus populaire pour apprendre les modèles de mélange (c'est à dire trouver θ_k et π_k) est l'algorithme espérance-maximisation (EM) [37]. Pour un nombre fixe d'éléments, l'algorithme EM est de nature itérative et comporte deux étapes à chaque itération. L'algorithme commence avec une estimation des paramètres initiaux. Souvent, cette estimation nécessite un regroupement automatique des données, comme une approche hiérarchique simple. La première étape de chaque itération (l'étape d'espérance) implique le calcul de l'espérance de la vraisemblance en tenant compte des dernières variables observées et des estimations des paramètres actuels. La seconde étape, l'étape de maximisation, estime le maximum de vraisemblance des paramètres en maximisant la vraisemblance trouvée à l'étape précédente. Le processus alterne entre ces deux étapes jusqu'à convergence.

Cependant, même si on considère généralement que l'algorithme EM converge de façon linéaire, la faible vitesse de convergence de l'algorithme EM a également été souligné comme le problème pratique le plus important. Par exemple, lorsque les données d'apprentissage sont composées d'un mélange de populations peu séparées, la vitesse de convergence de l'algorithme EM devient extrêmement lente [124, 117].

1.4.2 SOM : U*C

SOM : U*C [149] est une méthode qui utilise une carte SOM de très grande taille (plus de 4000 prototypes). Le nombre de prototypes de cette carte peut être beaucoup plus grand que le nombre de données, l'objectif étant d'obtenir une carte en deux dimensions de la structure des données en perdant le moins d'information possible. A partir de cette carte il est facile d'estimer la densité autour de chaque prototype (Matrice P : le nombre de données présentes dans un disque de rayon fixe) ainsi que les distances entre chaque prototype (Matrice M). A partir de ces informations il est possible de déterminer les zones de faible densité autour des prototypes et/ou les zones où il n'y a pas ou peu de prototypes, de façon à définir les limites entre les groupes (Matrice U*). La Matrice U* représente, pour chaque neurone, la proportion de neurones de plus grande densité multipliée par la distance moyenne entre le neurone et ses voisins. Cette valeur est proche de zéro dans les zones de forte densité (centre des groupes) et proche de la distance moyenne dans les zones de faible densité (bordure des groupes). Il est nécessaire de prendre en compte à la fois la distance et la densité puisque les zones vides de données sont aussi vides de prototypes et la densité seule ne permet pas de détecter ces zones. De la même manière, si des groupes sont en contact, les prototypes des deux groupes peuvent être très proches les uns des autres, bien que la frontière soit de faible densité.

L'algorithme SOM : U^*C est le suivant :

1) **Phase d'apprentissage** :
   - Appliquer SOM sur les données à partir d'une carte de très grande taille (ESOM).
   - Calculer les Matrices P, U et U^* à partir de la carte et des données.
2) **Phase d'immersion** :
   - A partir d'un neurone n suivre un gradient descendant sur la matrice U jusqu'à un minimum local (neurone u).

- A partir d'un neurone u suivre un gradient ascendant sur la matrice P jusqu'à un maximum local (neurone p).
- $I = I \cup \{p\}$; $Immersion(n) = p$.

3) **Phase d'assignation** :
   - Calculer les « watersheds » (voir Figure 1.7 et [154]) sur la matrice U^*, de façon à déterminer les frontières entre les groupes.
   - Donner un numéro de cluster $(C_1, ..., C_K)$ à chaque neurone de I.
   - Assigner à chaque donnée le neurone n le plus sensible à cette donnée et le numéro de cluster Cj tel que $Immersion(n) \in C_j$.

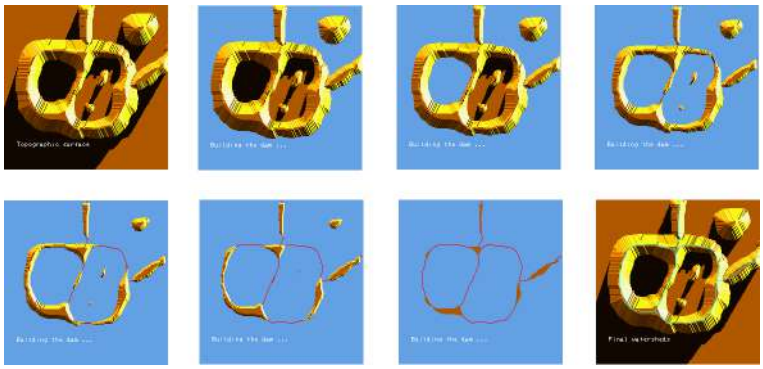


FIGURE 1.7 – **Principe des Watersheds** : la matrice U^* est représentée sous la forme d'une surface topologique. Les zones de faible densité sont représentées en relief. Les bassins (zones à forte densité) sont virtuellement remplis d'eau. Les Watersheds sont les lignes de contacts entre l'eau issue des différents bassins, lorsque le niveau d'eau recouvre tout le relief.

Sur des jeux de données artificielles proposées par l'auteur, les performances de classification de cet algorithme sont excellentes. Par contre, il est extrêmement coûteux en temps de calcul à la fois lors de la phase d'apprentissage (étant donnée la taille de la carte) et lors du calcul des watersheds. De plus, l'algorithme est limité par la contrainte topologique en deux dimensions de la carte auto-organisatrice. Ainsi, dans certains cas, des neurones non voisins peuvent être associés à un ensemble de données similaires. On obtient donc une "déchirure" de la projection des données sur la carte. Ce phénomène est un problème courants dans les algorithmes à base de SOM [6].

1.4.3 DBSCAN

DBSCAN [44, 161] est une autre méthode de classification à partir de la densité, mais elle ne s'appuie pas sur un ensemble de prototypes et s'applique directement sur les données. On définit deux paramètres Eps et $MinPt$. Eps est le rayon qui va servir à

calculer la densité autour d'un point et $MinPt$ est le nombre de points minimum situé dans un rayon de Eps autour d'une donnée pour que l'on considère que cette donnée fait partie du centre d'un groupe. Un point P est directement atteignable à partir d'un point Q si il y a plus de $MinPt$ points dans un rayon de Eps autour de P et que Q fait partie de ces points. Un point P_1 est atteignable à partir de P_n si il existe une suite de points $P_2, ..., P_{n-1}$ telle que pour tout i, P_i est directement atteignable à partir de P_{i+1}. Si deux points sont atteignables à partir d'une même données, ils appartiennent au même cluster (voir Figure 1.8).

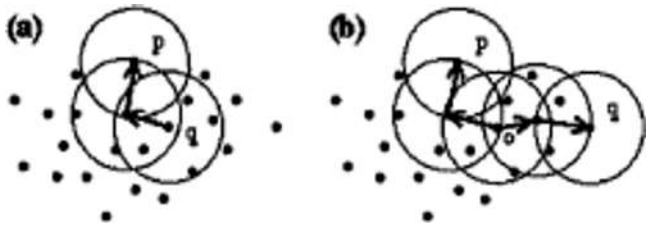


FIGURE 1.8 – **DBSCAN** : (a) p est atteignable à partir de q. (b) p et q appartiennent au même groupe

L'algorithme est très simple :

1) **Phase d'initialisation** :
   - Soit I l'ensemble des données.
2) **Phase d'assignation** :
   - Choisir une donnée $x^{(i)}$ au hasard parmi I.
   - Si il y a plus de $MinPt$ données dans un rayon de Eps autour de $x^{(i)}$ créer un nouveau cluster C_i , sinon enlever $x^{(i)}$ de I et passer à l'étape 3.
   - Chercher toutes les données atteignables à partir de $x^{(i)}$ et assigner ces données au cluster C_i .
   - Enlever de I les données assignées à Ci.
3) **Répéter la phase 2** jusqu'à ce que I soit vide.

Cet algorithme montre de bonnes performances sur des bases de données à deux dimensions avec bruit. Il est aussi nettement plus rapide que SOM : U*C. Par contre, il n'y a pas de réduction de dimensions, ce qui pose des problèmes de visualisation en dimensions supérieures à trois. De plus, le choix des paramètres est difficile sans

connaissance à priori de la structure des données, en particulier lorsqu'il y a de grandes variations de densité entre les groupes (voir [162] et [116]).

1.5 Conclusion

Ce panorama synthétique des différents algorithmes de classification non supervisée nous montre la variété des algorithmes proposés, mais aussi la difficulté du problème de partitionnement. En effet, la recherche de clusters naturels dans des jeux de données réelles, potentiellement de grande taille, impose un certain nombre de contraintes que l'algorithme utilisé doit satisfaire :

- Découverte de clusters de formes arbitraires.
- Faible complexité en temps de calcul.
- Résistance au bruit.
- Sélection automatique des paramètres, en particulier du nombre de clusters.
- Visualisation ou interprétation simple des résultats obtenus.

Aucun des algorithmes présentés dans ce chapitre ne satisfait simultanément toute ces contraintes. Beaucoup ont une complexité trop importante et quasiment aucun ne propose une estimation automatique du nombre de clusters. Les méthodes à deux niveaux présentent de nombreux avantages (faible complexité, résistance au bruit, éventuellement visualisation si on se base sur une SOM) mais souffrent à la fois de la perte d'information liée à la représentation des données au premier niveau et des défauts de l'algorithme de classification utilisé au second niveau.

Nous proposons donc dans le prochain chapitre de nouveaux algorithmes de classification, inspirées des méthodes à deux niveaux, qui sont capables de satisfaire simultanément toute les contraintes évoquées dans cette section.

Chapitre 2

Classification selon le voisinage et la densité

2.1 Introduction

Dans ce chapitre, nous proposons de nouveaux algorithmes de classification non-supervisée adaptées aux données vectorielles. Ces algorithmes sont inspirés des méthodes à deux niveaux et se basent sur l'apprentissage d'une carte auto-organisatrice. Cependant, contrairement aux algorithmes à deux niveaux classiques, ils sont capables d'apprendre simultanément les prototypes de la carte et sa segmentation à partir des données, de façon à éviter une perte d'information trop importante lors de l'étape de quantification des données. La première partie de ce chapitre présente un algorithme s'inspirant du Competitive Hebbian Learning pour regrouper les neurones d'une SOM à partir des données pendant son apprentissage. Cet algorithme utilise une estimation de la connectivité des prototypes à partir des distances entre les données et les prototypes, ce qui lui permet de détecter des clusters bien séparés sans a priori forts sur le nombre ou la structure des clusters. Dans une deuxième partie, nous proposons un algorithme qui utilise, en plus de la connectivité des prototypes, des informations sur la densité des données. Ce nouvel algorithme conserve les propriétés du premier. Il est en outre capable de détecter des clusters en contact ou très bruités, en détectant les zones de faible densité définissant les frontières entre ces clusters.

2.2 Classification selon le voisinage

Une des questions les plus importantes pour la plupart des applications réelles, aussi connue comme le "problème de sélection de modèle", est de déterminer un nombre approprié de groupes. Sans connaissances a priori il n'y a pas de moyen simple pour déterminer ce nombre [108, 47, 147, 39, 65]. L'objectif de ce travail est de fournir une approche de classification à deux niveaux simultanés utilisant une SOM, qui peut être appliquée à de grandes bases de données. La méthode proposée regroupe automatiquement les données, c'est-à-dire que le nombre de groupes est déterminé automatiquement pendant le processus d'apprentissage, i.e. aucune hypothèse a priori sur le nombre de groupes n'est exigée. Cette approche a été évaluée sur un jeu de problèmes fondamentaux pour la classification et montre d'excellents résultats comparés aux approches classiques.

Des travaux récents [144, 145] ont montré qu'il est possible d'assigner aux connexions topologiques d'une carte auto-organisatrice une valeur représentative de la distribution des données entre prototypes. Cette valeur, $Conn(u^*, u^{**})$, est le nombre de données pour lesquelles u^* et u^{**} sont les deux meilleurs représentants. De plus, puisqu'ici on

utilise une carte en deux dimensions, il est possible de visualiser ces connexions et d'obtenir des informations sur la structure générale des clusters entre eux.

Ces travaux proposent d'utiliser cette information comme un critère de validité de la segmentation obtenue avec une méthode à deux niveaux classique. Il nous semble néanmoins possible d'estimer cette information pendant l'apprentissage, selon le principe du CHL et de l'utiliser pour découvrir, automatiquement et en une seule étape, une classification des données.

Nous proposons donc une adaptation de l'algorithme CHL à une SOM, de façon à profiter des performances de classification du CHL et des performances de réduction de dimension et de visualisation d'une SOM, qui autorise des comparaisons directes entre clusters.

2.2.1 Un nouvel algorithme de classification à deux niveaux : S2L-SOM

Une SOM est composée d'un ensemble de prototypes connectés par des connexions topologiques. Ces connexions indiquent simplement les relations de voisinage entre prototypes sur la carte.

Avec S2L-SOM (Simultaneous 2-Level – Self-Organising Map), chaque connexion de voisinage est associée à une valeur réelle v qui indique la pertinence de la connexion entre deux neurones, c'est à dire la pertinence du couple connecté pour la représentation locale des données. Étant donnée la contrainte d'organisation de la carte, les deux meilleurs représentants de chaque donnée (les neurones les plus sensibles) doivent être reliés par une connexion topologique. Cette connexion sera « récompensée » par une augmentation de sa valeur, d'autant plus importante que l'apprentissage est avancé et que les prototypes représentent bien les données, alors que toutes les autres connexions issues du meilleur représentant seront « punies » par une diminution de leurs valeurs. Les informations mises à jours en fin d'apprentissage sont favorisées (c'est le rôle de le fonction $r(t)$) pour tenir compte de l'évolution de la validité de la structure de la carte au cours du temps pendant l'apprentissage. Ainsi, à la fin de l'apprentissage, un ensemble de prototypes inter-connectés (avec des connexions de valeurs supérieures à zéro) sera représentatif d'un sous-groupe pertinent de l'ensemble des données. [101] a montré que le graphe généré de cette manière préserve la topologie de façon optimale. En particulier chaque arc de ce graphe suit la triangulation de Delaunay correspondant aux vecteurs de référence.

La fonction de coût de S2L-SOM devient une combinaison de deux fonctions :

$$\widetilde{R}(w,v) = \widetilde{R}(w) + \widetilde{R}(s)$$

– Le coût lié à l'estimation des prototypes :

$$\widetilde{R}(w) = \sum_{k=1}^{N}\sum_{j=1}^{M} K_{ju^*(x^{(k)})}\|w_j - x^{(k)}\|^2$$

– Le coût lié à l'estimation des valeurs de connexions :

$$\widetilde{R}(v) = \sum_{k=1}^{N}\sum_{i,j=1}^{M}\left[v_{ij} - \mathbb{1}_{\{w_i,w_j \text{ bmus de } x^{(k)}\}}\right]^2$$

L'algorithme S2L-SOM est le suivant :

1) **Phase d'initialisation** :
   – Définir la topologie de la carte.
   – Initialiser aléatoirement tous les prototypes $w_j = (w_{0j}, ..., w_{nj})$ pour chaque neurone j.
   – Initialiser les connexions ν entre chaque couple de neurones i et j :

$$\forall i,j \quad \nu_{ij} = 0$$

2) **Phase de compétition** :
   – Présenter une donnée $x^{(k)}$ choisie aléatoirement.
   – Parmi les M neurones, choisir les deux meilleurs $u^*(x^{(k)})$ et $u^{**}(x^{(k)})$ pour représenter cette donnée :

$$u^*(x^{(k)}) = \underset{1\leq i\leq M}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$
$$u^{**}(x^{(k)}) = \underset{1\leq i\leq M,\quad i\neq u^*}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

   – Augmenter la valeur de la connexion entre $u^*(x^{(k)})$ et $u^{**}(x^{(k)})$ et diminuer les valeurs des autres connexions issues de $u^*(x^{(k)})$:

$$\nu_{u^*u^{**}}(t) = \nu_{u^*u^{**}}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*u^{**}}(t-1) - 1\right)$$
$$\nu_{u^*i}(t) = \nu_{u^*i}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*i}(t-1)\right) \quad \forall i \text{ voisin de } u^*$$

   Avec :

$$r(t) = \frac{1}{1 + e^{-\left(\frac{t}{t_{max}}\right)}}$$

   t_{max} étant le nombre max d'itérations.

3) **Phase d'adaptation** :
   – Mettre à jour les prototypes w_j de chaque neurone j selon la règle :

$$w_j(t) = w_j(t-1) - \varepsilon(t) K_{ju^*(x^{(k)})}(w_j(t-1) - x^{(k)})$$

4) **Répéter les phases 2 et 3** jusqu'à ce que les mises à jours des prototypes soient négligeables.

Cet algorithme est proche d'une CHL, mais ici seuls les neurones voisins sur la carte peuvent être connectés, ce qui conserve la topologie en deux dimensions de la carte et permet une réduction de dimensions et une visualisation simple de la structure des données. Par ailleurs, l'utilisation d'une valeur de récompense plutôt qu'un âge donne une information sur la représentativité locale des deux neurones connectés. De plus, puisque toutes les connexions sont conservées, on garde une information sur la structure générale des données et des clusters entre eux. Il faut noter que le résultat final dépend en partie de l'ordre de présentation des données (cet ordre est aléatoire) et peut donc varier légèrement d'une exécution à l'autre.

Nous avons évalué les performances de cet algorithme sur un ensemble de jeux de données artificielles présentant des difficultés pour la classification.

2.2.2 Validation

2.2.2.1 Description des bases de données utilisées

L'algorithme à été testé sur dix jeux de données artificielles (Table 2.1). Les bases de données «Hepta», «Chainlink», «Atom», «Engytime», «Wingnut» et «Twodiamonds» proviennent du Fundamental Clustering Problem Suite (FCPS, [149]), nous avons généré aussi quatre autres bases de données intéressantes («Rings», «Spirals», «Highdim» et «Random»).

«Rings» est composée de K=3 groupes en D=2 dimensions non linéairement séparables et de densités et variances différentes : un anneau de rayon 1 pour 700 points (forte densité), un anneau de rayon 3 pour 300 points (faible densité) et un anneau de rayon 5 pour 1500 point (densité moyenne). «Highdim» est constitué de 9 groupes de 100 points chacun bien séparés dans un espace à 15 dimensions. «Random» est un tirage aléatoire de N=1000 points dans 3 dimensions. Enfin «Spirals» est constitué de deux spirales parallèles de 1000 points chacune dans un anneaux de 3000 points. La densité des points dans les spirales diminue avec le rayon.

TABLE 2.1 – Description des données de test

Nom	N	D	K	Problème spécifique
Highdim	900	15	9	Grande dimension
Spirals	5000	2	3	Non linéairement séparables, gradient de densité
Rings	2500	2	3	Non linéairement séparables, différentes densités
Twodiamonds	800	2	2	Groupes en contact
Chainlink	1000	3	2	Non linéairement séparables
Atom	800	3	2	Non linéairement séparables, différentes densités
Hepta	212	3	7	Différentes densités
Random	1000	3	1	Aucune structure
Wingnut	1070	2	2	Gradients opposés de densité
Engytime	4096	2	2	Groupes définis par la densité

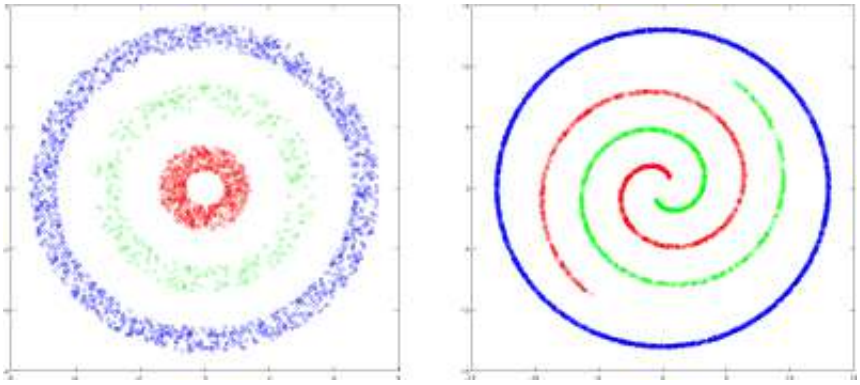


FIGURE 2.1 – Données «Rings» et «Spirals»

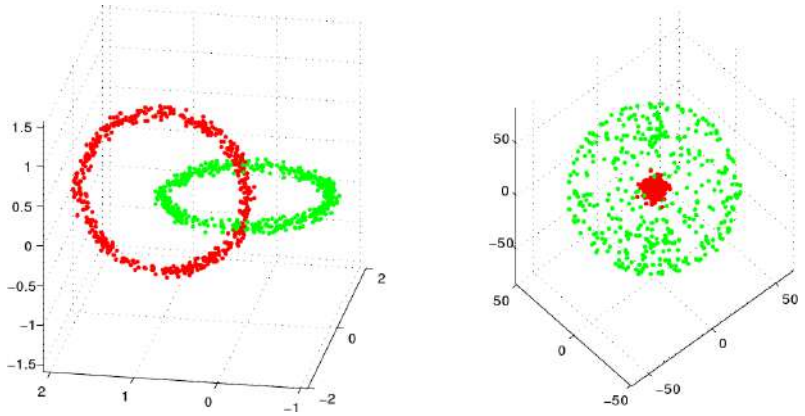


FIGURE 2.2 – Données «Chainlink» et «Atom»

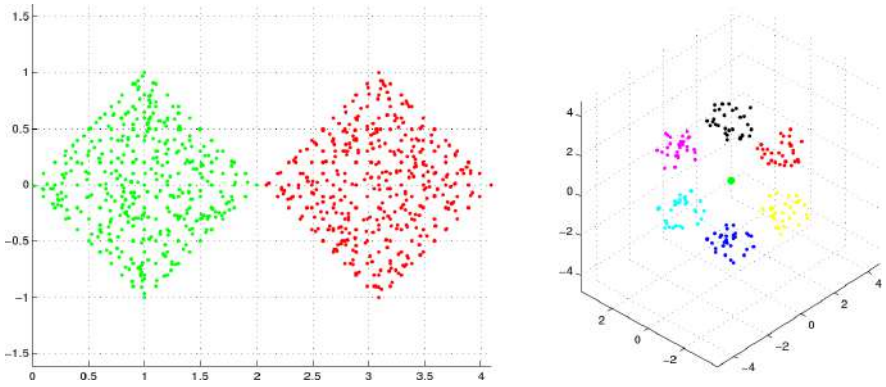


FIGURE 2.3 – Données «Diamonds» et «Hepta»

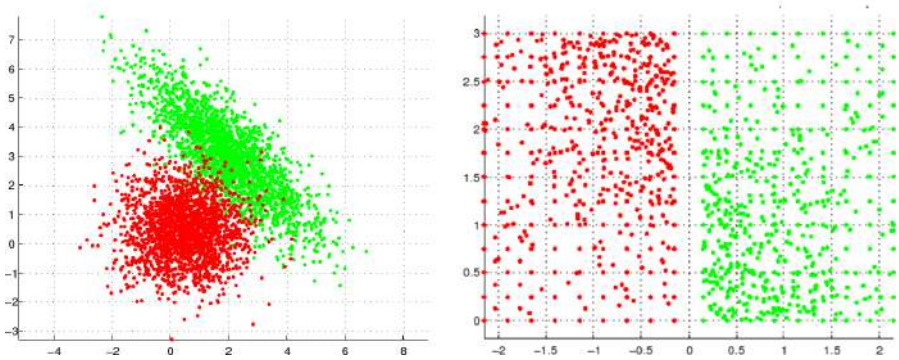


FIGURE 2.4 – Données «Engytime» et «Wingnut»

2.2.2.2 Protocole expérimental

Nous avons comparé les performances de S2L-SOM par rapport aux performances des méthodes classiques à un et deux niveaux. Les algorithmes de comparaison choisis

sont K-Moyennes [106] et CAH-SingleLinkage [78, 142] appliqués directement sur les données ou bien sur les prototypes de la carte après apprentissage par S2L-SOM (de cette façon, chaque méthode à deux niveaux effectue une segmentation à partir de la même carte). On attribue alors à chaque donnée le numéro de cluster de son prototype le plus proche. L'indice de Davis-Bouldin [35] est utilisé pour déterminer le meilleur découpage des arbres (CAH) ou le nombre optimal de centroïdes (K-Moyennes). S2L-SOM détermine automatiquement le nombre de clusters et n'a pas besoin d'utiliser un indice.

La qualité de la segmentation peut être évaluée à partir d'indices internes (indices de Davies-Bouldin [35] ou Calinski-Harabasz [25] par exemple) et externes (indices de Jacquard [77] , Rand [123], ...) fréquemment utilisés.

Les indices internes évaluent la qualité d'une segmentation par l'analyse de sa structure. Ainsi, les données d'un même cluster doivent être similaires, contrairement aux données appartenant à des clusters différents.

Davies-Bouldin :

$$DB(K) = \frac{1}{K} \sum_{i=1}^{K} \frac{s_i + s_j}{\parallel w_i - w_j \parallel^2}$$

Avec K le nombre de clusters, s_i la variance interne du cluster i de centroïde w_i. L'indice est d'autant plus petit que la segmentation est de bonne qualité.

Calinski-Harabasz :

$$CH(K) = \frac{B/(k-1)}{W/(N-K)}$$

Avec K le nombre de clusters, N le nombre total de donnée, W et B les variances inter- et intra-cluster. L'indice est d'autant plus grand que la segmentation est de bonne qualité.

Les indices internes classiques ne sont cependant pas adaptés pour évaluer une segmentation composée de clusters non linéairement séparables. On peut néanmoins utiliser des indices externes, lorsque la segmentation souhaitée est connue, en particulier sur des jeux de données de test de petites dimensions. Il s'agit de comparaisons entre la segmentation proposée par l'algorithme et une segmentation souhaitée.

$$Rand = \frac{a_{11} + a_{00}}{a_{11} + a_{01} + a_{10} + a_{00}}$$

$$Jaccard = \frac{a_{11}}{a_{11} + a_{01} + a_{10}}$$

Avec a_{11} le nombre de paires de données correctement classées dans le même cluster, a_{01} le nombre de paires de données incorrectement classées dans des clusters différents, a_{10} le nombre de paires de données incorrectement classées dans le même cluster et a_{00} le nombre de paires de données correctement classées dans des clusters différents.

Nous avons utilisé pour cette étude l'indice externe de Jaccard, qui est le plus utilisé dans ce domaine (Table 2.2 et Figure 2.5). Nous avons calculé l'indice moyen obtenu sur une centaine d'executions de l'algorithme.

Nous avons aussi évalué la stabilité des différents algorithmes (Table 2.3 et Figure 2.6). La stabilité d'une classification est de plus en plus utilisée comme un indice fiable de sa validité [53, 95, 12, 126, 5]. Nous avons tiré aléatoirement pour chaque base de données deux sous-échantillons (80% des points à chaque fois), sur lesquels nous avons appliqué un algorithme de classification. La différence entre les deux segmentation obtenues a ensuite été mesurée par l'indice de Jaccard. Le processus est répété un grand nombre de fois et la moyenne de l'indice est considéré comme une estimation fiable de la stabilité de l'algorithme [12].

2.2.2.3 Résultats

Les classifications obtenues avec S2L-SOM sont trés stables et d'excellentes qualités pour les données regroupées selon un ensemble de distribution gaussienne, quel que soit la dimension (« Hepta » et « HighDim »), mais aussi dans les cas où les groupes sont de formes arbitraires en deux dimensions (« Rings » et « Spirals ») et lorsque les données ne sont pas structurées (« Random »). Il faut noter que seul S2L-SOM peut détecter l'absence de structure, puisque les méthodes classiques déterminent le nombre de groupe optimal à l'aide d'un indice de qualité qui ne se calcule qu'a partir de deux groupes minimum. Les algorithmes classiques montrent aussi de bonnes performances pour les données regroupées selon un ensemble de distributions gaussiennes (Figure 2.7), mais leurs performances sont très mauvaises lorsque les groupes sont de formes arbitraires ou que les données ne sont pas structurées (Figures 2.8 et 2.9).

Les représentations visuelles des cartes après apprentissage sont composées d'un ensemble d'hexagones qui représentent les prototypes. La distance entre deux hexagones est représentative du voisinage entre les prototypes correspondants. Les prototypes

TABLE 2.2 – Comparaison de la qualité de la segmentation avec l'indice de Jaccard moyen (arrondi au centième) selon les bases de données et les méthodes utilisées.

Données	CAH	K-Moyenne	SOM+CAH	SOM+K-Moyenne	**S2L-SOM**
HighDim	100%	82%	43%	100%	**100%**
Spirals	91%	11%	44%	12%	**100%**
Rings	97%	26%	47%	32%	**100%**
Chainlink	72%	14%	50%	17%	**100%**
Atom	87%	56%	44%	57%	**100%**
Hepta	100%	100%	76%	100%	**100%**
Random	97%	07%	88%	07%	**100%**
Twodiamonds	97%	100%	100%	100%	**89%**
Wingnut	99%	59%	90%	59%	**50%**
Engytime	50%	32%	47%	18%	**50%**
Moyenne	89%	49%	63%	50%	**89%**

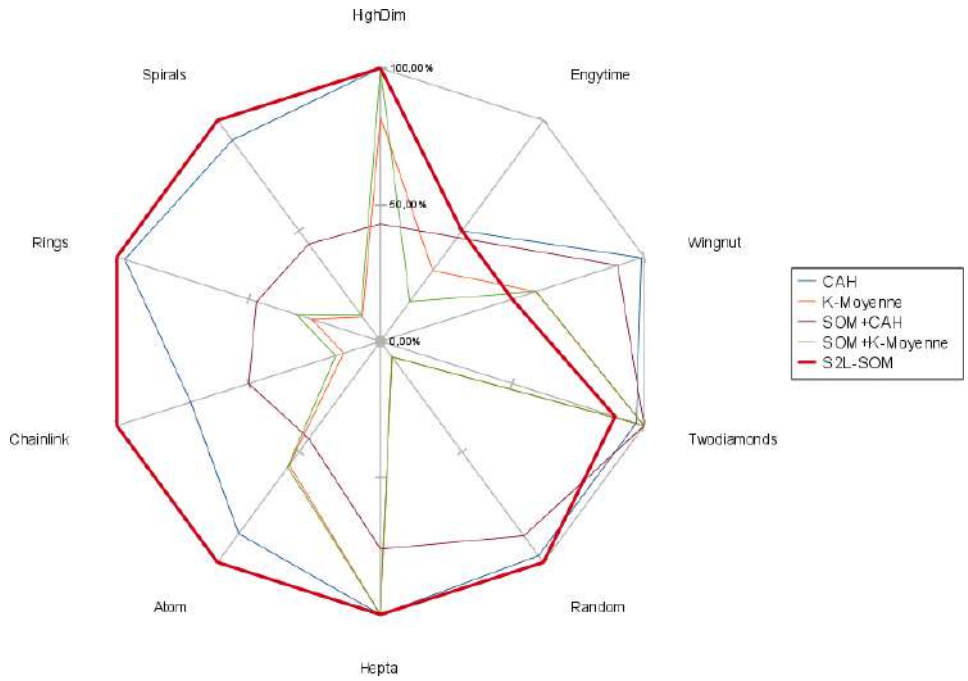


FIGURE 2.5 – Comparaison de la qualité de la segmentation avec l'indice de Jaccard selon les bases de données et les méthodes utilisées.

voisins sur la carte sont représentés par des hexagones en contact et ceux représentés par des hexagones de même couleur appartiennent à un même cluster. Certains prototypes non représentatifs des données peuvent ne pas être connectés à d'autres par S2L-SOM, dans ce cas les hexagones correspondants sont représentés en gris. C'est une information

TABLE 2.3 – Comparaison de la stabilité de la classification selon les bases de données et les méthodes utilisées.

Données	CAH	K-Moyenne	SOM+CAH	SOM+K-Moyenne	**S2L-SOM**
HighDim	100%	84%	56%	90%	**100%**
Spirals	72%	75%	99%	46%	**97%**
Rings	100%	72%	90%	66%	**100%**
Chainlink	57%	68%	81%	50%	**82%**
Atom	88%	78%	80%	85%	**87%**
Hepta	100%	94%	71%	98%	**99%**
Random	97%	21%	65%	13%	**100%**
Twodiamonds	66%	94%	100%	99%	**84%**
Wingnut	95%	59%	51%	45%	**100%**
Engytime	99%	27%	90%	38%	**100%**
Moyenne	87%	67%	78%	63%	**95%**

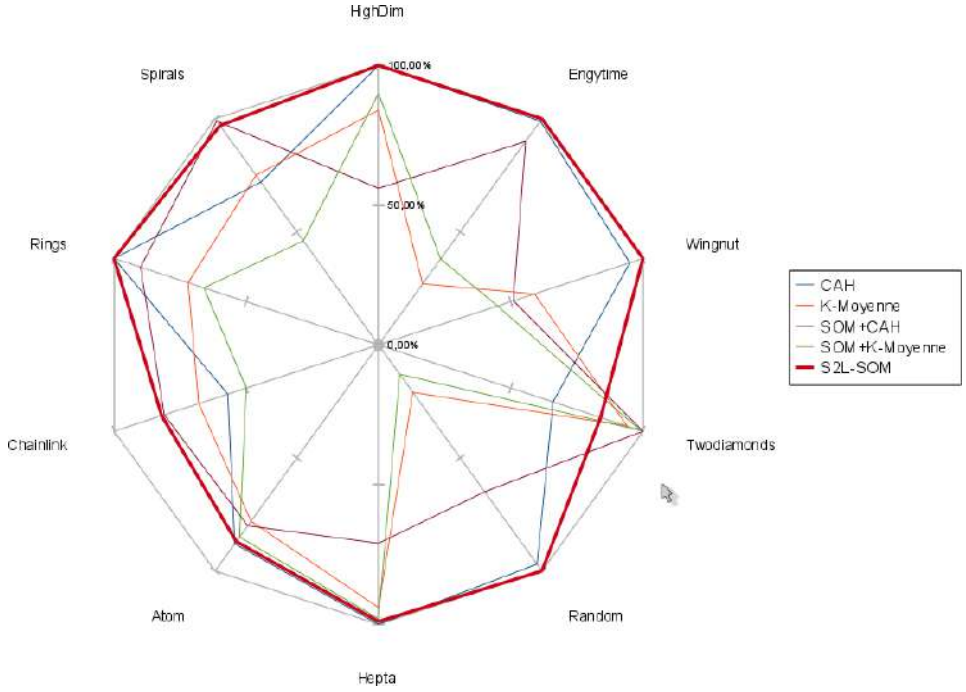


FIGURE 2.6 – Comparaison de la stabilité de la classification selon les bases de données et les méthodes utilisées.

que seul S2L-SOM est capable de donner.

Lorsque les données ne sont pas linéairement séparables dans des dimensions supérieures à 2 (« Atom » et « Chainlink »), l'algorithme est limité par la contrainte to-

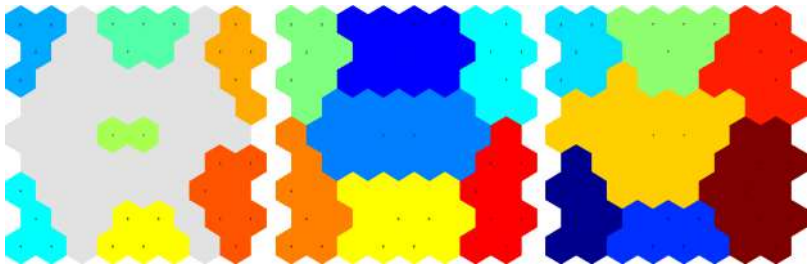


FIGURE 2.7 – Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Hepta»

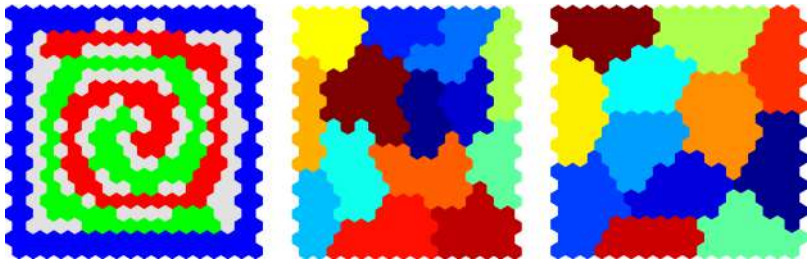


FIGURE 2.8 – Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Spirals»

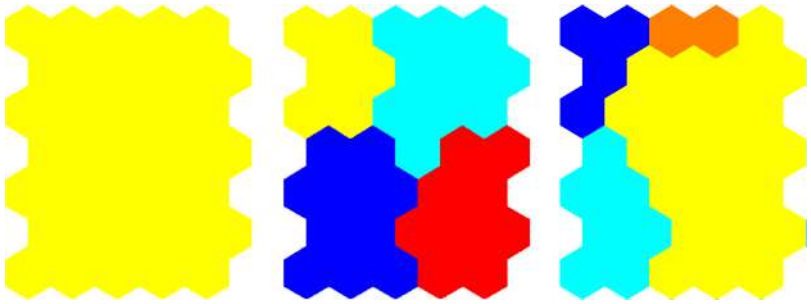


FIGURE 2.9 – Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Random»

pologique en deux dimensions de la carte auto-organisatrice. En particulier, on obtient parfois la séparation d'un cluster en deux pour conserver cette topologie. Cependant, dans ces cas certains neurones non voisins sont associés à une valeur v non nulle. Ainsi, un cluster peut être séparé en deux ensembles de neurones non adjacents sur la carte tout en étant considéré comme un unique cluster et représenté comme tel (voir Figure 2.10 pour un exemple).

En fait, deux types de défauts topologiques courants dans les SOM (cf. [6]) peuvent être détectés par S2L-SOM :

– **Les recollements** : deux neurones sont voisins sur la carte mais ne représentent pas les mêmes données. Les recollements se visualisent sur la carte comme des neurones voisins appartenant à des groupes différents (donc de différentes couleurs).
– **Les déchirures** : deux neurones sont éloignés sur la carte mais représentent les mêmes données. Les déchirures se visualisent sur la carte par des ensembles séparés (non connexes) de neurones appartenant au même groupe (donc par des neurones de même couleur n'étant pas en contact).

Ainsi, S2L-SOM est capable de « réparer » implicitement les défauts topologiques de la SOM en produisant des clusters qui tiennent compte de la topologie d'origine. Cela permet, en une seule phase, d'effectuer un clustering des données et de visualiser les clusters par une projection qui tient compte de leur topologie.

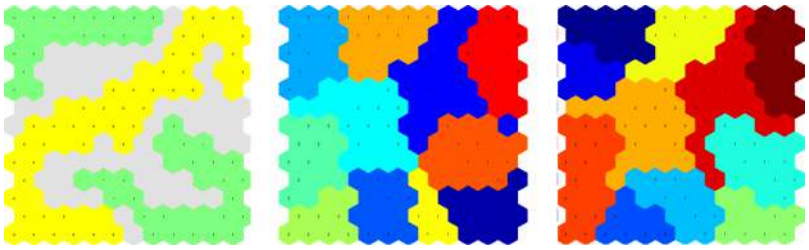


FIGURE 2.10 – Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Chainlink»

Par contre, S2L-SOM n'est pas capable de séparer des groupes en contact, dont la frontière est définie par une diminution locale de la densité (« Engytime » et « Wingnut »). En effet, ce contact favorise la création et l'augmentation de la valeur des connexions entre les deux groupes (Figure 2.11). C'est aussi ce qui explique la baisse de qualité et de stabilité pour « Twodiamonds ». Par contre la stabilité de « Engytime » et « Wingnut » est très bonne puisque l'algorithme ne trouve jamais de séparation entre les deux groupes.

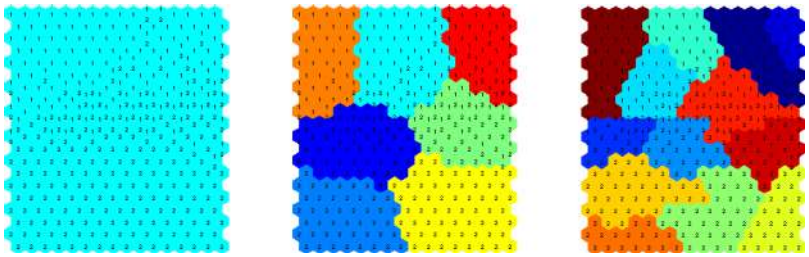


FIGURE 2.11 – Cartes auto-organisatrices segmentées par, de gauche à droite, S2L-SOM, SOM+K-Moyenne et SOM+CAH à partir des données «Engytime»

2.2.3 Conclusions

Dans cette première partie du chapitre, nous avons proposé une méthode de classification à deux niveaux simultanés. On utilise une SOM comme technique de réduction de dimensions et effectue en parallèle une classification optimisée. Les performances de cette méthode ont été évaluées à partir de tests sur une série de problèmes fondamentaux pour la classification, et comparées aux méthodes à deux niveaux classiques s'appuyant sur CAH ou K-Moyennes. Les résultats expérimentaux démontrent que l'algorithme proposé produit une classification de meilleure qualité que les approches classiques. Ils montrent aussi que le grand avantage de l'algorithme S2L-SOM est qu'il n'est pas limité aux groupes de formes convexes, mais est capable d'identifier des groupes de formes arbitraires. Pour finir, le nombre de groupes est déterminé automatiquement dans notre approche pendant l'apprentissage, c'est-à-dire qu'aucun a priori sur ce nombre n'est requis.

Cependant, cette méthode ne peut fonctionner que si les clusters sont suffisamment séparés dans l'espace de données. En effet, des groupes qui se touchent ne sont définis que par une diminution de la densité dans la zone de contact, ce qui ne peut pas être détecté par S2L-SOM. Pour corriger ce problème nous proposons une deuxième version de S2L-SOM qui apprend aussi une estimation de la densité autour des prototypes pendant l'apprentissage et qui se sert de cette estimation pour séparer les groupes en contact.

2.3 Classification selon la densité

L'objectif de cette deuxième partie est de proposer une méthode de classification à deux niveaux simultanés qui, comme S2L-SOM, s'appuie sur l'apprentissage d'une SOM tout en conservant le maximum d'information sur la structure des données. Cependant, nous proposons ici une approche basée à la fois sur la distance et sur la détection des modes de densités [130], s'appuyant sur le fait qu'un regroupement de données peut être défini comme une région de l'espace de représentation localement dense en données, entourée par une région de faible densité [44, 161, 149, 9, 115, 116]. Cette approche est particulièrement efficace lorsque les groupes se touchent ou en présence de bruit. Elle est aussi efficace pour la détection des groupes non convexes. De plus, la méthode proposée regroupe automatiquement les données, c'est-à-dire que le nombre de groupes est déterminé automatiquement pendant le processus d'apprentissage, aucune hypothèse a priori sur le nombre de groupes n'est exigée. Cette approche a été évaluée sur un jeu

de problèmes fondamentaux représentant différentes difficultés pour la classification et montre d'excellents résultats comparé aux approches classiques.

2.3.1 Algorithmes

Nous proposons donc ici un nouvel algorithme de classification topographique non-supervisée, DS2L-SOM (Density-based Simultaneous 2-Level – SOM), qui apprend simultanément les prototypes (référents) d'une carte auto-organisatrice et sa segmentation en utilisant à la fois des informations sur les distances et la densité des données. L'algorithme estime pendant l'apprentissage la densité locale des données, afin de détecter les fluctuations de densité qui caractérisent les frontières entre les groupes de données.

2.3.1.1 Apprentissage de la carte

L'algorithme DS2L-SOM est une adaptation de l'algorithme S2L-SOM. Avec S2L-SOM, chaque connexion de voisinage est associée à une valeur réelle v qui indique la pertinence de la connexion entre deux neurones. Ainsi, à la fin de l'apprentissage, un ensemble de prototypes inter-connectés sera une bonne représentation de sous-groupes bien séparés de l'ensemble des données. Dans l'algorithme DS2L-SOM, nous proposons en outre d'associer à chaque unité j une estimation de la densité locale des données D_j, de manière à détecter les fluctuations de densité qui définissent les frontières entre groupes en contact. Pour chaque donnée, cette valeur de densité sera augmentée pour tous les prototypes, en fonction de la distance euclidienne entre le prototype et les données. On remarque que, dans l'algorithme DS2L-SOM, l'estimation de la densité locale des données est faite pendant la formation de la carte, c'est-à-dire qu'il n'est pas nécessaire de conserver les données en mémoire.

La fonction de coût que l'on cherche à minimiser dans DS2L-SOM est en fait une combinaison de trois fonctions de coût :

$$\widetilde{R}(w, D, v) = \widetilde{R}(w) + \widetilde{R}(D) + \widetilde{R}(v)$$

– Le coût lié à l'estimation des prototypes :

$$\widetilde{R}(w) = \sum_{k=1}^{N} \sum_{j=1}^{M} K_{ju^*(x^{(k)})} \|w_j - x^{(k)}\|^2$$

– Le coût lié à l'estimation des densités :

$$\widetilde{R}(D) = \sum_{k=1}^{N} \sum_{j=1}^{M} \left[D_j - e^{\frac{\|w_j - x^{(k)}\|^2}{2\sigma^2}} \right]^2$$

– Le coût lié à l'estimation des valeurs de connexions :

$$\widetilde{R}(v) = \sum_{k=1}^{N} \sum_{i,j=1}^{M} \left[v_{ij} - \mathbb{1}_{\{w_i, w_j \text{ bmus de } x^{(k)}\}} \right]^2$$

L'algorithme d'apprentissage de DS2L-SOM procède en trois phases :

Entrées :
– Les données $X = \{x^{(k)}\}_{k=1..N}$.
– SOM avec M prototypes $\{w_j\}_{j=1..M}$.
– t_{max} : le nombre maximum d'itérations.
Sorties :
– Une partition $P = \{C_i\}_{i=1..L}$ d'ensembles d'unités inter-connectés.
– Des valeurs de densité $\{D_j\}_{j=1..M}$ associées à chaque unité.

1) **Phase d'initialisation** :
   – Initialiser toutes les valeurs des connections de voisinage v à zéro.
   – Initialiser toutes les valeurs de densité D_j des unités à zéro.

2) **Phase de compétition** :
   – Présenter une donnée $x^{(k)}$ choisie aléatoirement.
   – Trouver le premier BMU u^* et le second BMU u^{**} :

$$u^*(x^{(k)}) = \underset{1 \leq i \leq M}{Argmin} \parallel x^{(k)} - w_j \parallel^2$$

   et

$$u^{**}(x^{(k)}) = \underset{1 \leq i \leq M,\ i \neq u^*}{Argmin} \parallel x^{(k)} - w_j \parallel^2$$

3) **Phase d'adaptation** :
   – Mettre à jour les prototypes w_j de chaque unité i selon la règle d'adaptation ci-dessous et le pas d'apprentissage $\varepsilon(t)$:

$$w_j(t) = w_j(t-1) - \varepsilon(t) K_{j,u^*(x^{(k)})}(w_j(t-1) - x^{(k)})$$

– Mettre à jours les valeurs de densités D_j associées à chaque unité j :

$$D_j(t) = D_j(t-1) - \varepsilon(t)r(t)\left(D_j(t-1) - e^{-\frac{\|x^{(k)}-w_j(t)\|^2}{2\sigma^2}}\right)$$

avec

$$r(t) = \frac{1}{1+e^{\left(-\frac{t}{t_{max}}\right)}}$$

– Actualiser les valeurs des connexions de voisinage v, conformément aux règles suivantes :

$$\nu_{u^*u^{**}}(t) = \nu_{u^*u^{**}}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*u^{**}}(t-1) - 1\right)$$
$$\nu_{u^*i}(t) = \nu_{u^*i}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*i}(t-1)\right) \quad \forall i \text{ voisin de } u^*$$

Avec :

$$r(t) = \frac{1}{1+e^{-\left(\frac{t}{t_{max}}\right)}}$$

4) **Répéter les étapes 2 et 3** jusqu'à ce que $t = t_{max}$

5) **Extraire tous les ensembles d'unités connectées** :
Soit $P = \{C_i\}_{i=1..L}$ les L ensembles d'unités interconnectées tel que $v > 0$ (voir Figure 2.13(b)).

6) **Retourner** $P = \{C_i\}_{i=1..L}$ et $\{D_j\}_{j=1..M}$.

La méthode proposée pour estimer les modes de densité est très similaire à celle proposé par [116]. Il a été montré que lorsque le nombre de points de données tend vers l'infini, l'estimateur D converge asymptotiquement vers la vraie fonction de densité [139]. Le choix du paramètre σ est important pour de bons résultats, mais sa valeur optimale est difficile à calculer et coûteuse en temps de calcul (voir [128]). Si σ est trop grand, toutes les données vont influencer la densité de tous les prototypes, les prototypes proches sont alors associés à des densités similaires, induisant une diminution de la précision de l'estimation. Si σ est trop petit, une grande proportion des données (les plus éloignées des prototypes) n'influenceront pas la densité des prototypes, ce qui induit une perte d'information. Une heuristique qui nous semble pertinente et qui donne de bons résultats est de définir σ comme la distance moyenne entre un prototype et son voisin le plus proche.

2.3.1.2 Raffinement de la segmentation

A la fin du processus d'apprentissage, nous utilisons un algorithme de raffinement qui utilise les informations de connexions et de densités pour détecter les groupes. Les neurones qui sont reliés par des connexions de voisinage tels que $v > 0$ définissent des groupes bien distincts. Nous utilisons alors une méthode de "Watersheds" [154] sur la carte de la densité de chacun de ces groupes pour détecter les zones de faible densité à l'intérieur des groupes bien séparés, de façon à caractériser les sous-groupes définis par la densité. Nous utilisons pour chaque paire de sous-groupes adjacents un indice "densité-dépendant" [161] pour déterminer si une zone de faible densité est un indicateur fiable de la structure des données, ou si elle doit être considérée comme une fluctuation aléatoire de la densité (voir Figure 2.12).

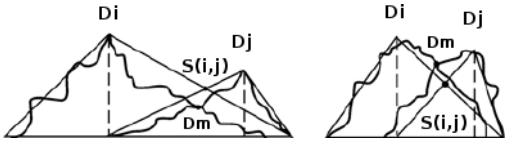


FIGURE 2.12 – Principe du calcul du seuil. À gauche : la densité minimale Dm entre les deux sous-groupes i et j de densité maximale D_i et D_j est inférieure au seuil $S(i,j)$, les deux sous-groupes ne sont pas fusionnés. À droite : la densité Dm est supérieure au seuil $S(i,j)$, les deux sous-groupes sont fusionnés.

Cette division est très rapide en raison du faible nombre de prototypes. L'utilisation combinée de ces deux types de définition des groupes permet d'obtenir de bons résultats en dépit du faible nombre de prototypes de la carte.

Entrées : $P = \{C_i\}_{i=1..L}$ et $\{D_j\}_{j=1..M}$.

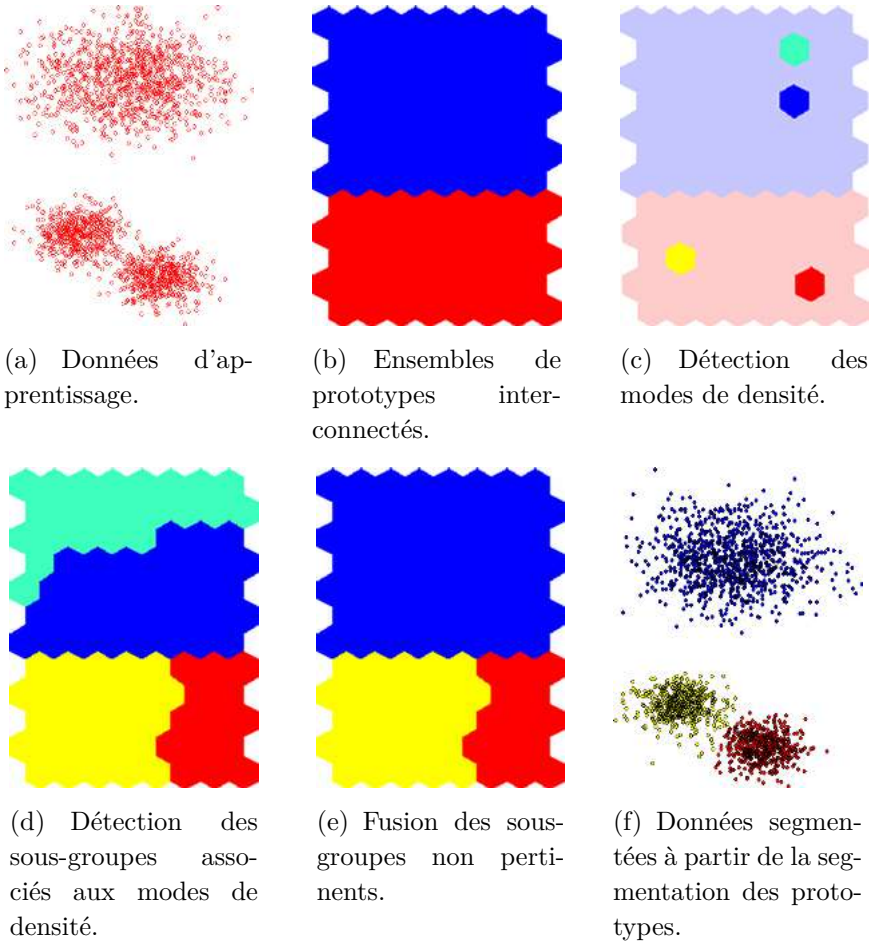
Sortie : Les groupes raffinés (les *clusters*).

1) **Pour chaque groupe** $C_k \in P$ **faire** :
   - Déterminer l'ensemble $M(C_k)$ des maximums locaux de densité (les modes de densité, voir Figure 2.13(c)) :

$$M(C_k) = \{i \in C_k \mid D_j \geq D_j, \forall j \text{ voisin de } i\}$$

   - Calculer la matrice des seuils :

$$S = [S(i,j)]_{i,j=1...|M(C_k)|}$$



(a) Données d'apprentissage.

(b) Ensembles de prototypes interconnectés.

(c) Détection des modes de densité.

(d) Détection des sous-groupes associés aux modes de densité.

(e) Fusion des sous-groupes non pertinents.

(f) Données segmentées à partir de la segmentation des prototypes.

FIGURE 2.13 – Exemple de déroulement des différentes étapes de l'algorithme de raffinement

avec

$$S(i,j) = \left(\frac{1}{D_i} + \frac{1}{D_j}\right)^{-1}$$

- Pour chaque neurone $i \in C_k$, étiqueter i avec un élément $label(i)$ de $M(C_k)$, selon un gradient ascendant de densité le long des connexions topologiques. Chaque étiquette représente un groupe (voir Figure 2.13(d)).
- Pour chaque paire de neurones voisins (i,j) dans C_k, si $label(i) \neq label(j)$ et si $D_i > S(i,j)$ et $D_j > S(i,j)$ alors fusionner les deux groupes (voir Figure 2.13(e)).

2) **Retourner les groupes raffinés** (les *clusters*).

2.3.2 Validation

2.3.2.1 Protocole expérimental

Nous avons comparé les performances de DS2L-SOM en terme de qualité de la segmentation et de stabilité par rapport aux performances de S2L-SOM et des méthodes classiques à deux niveaux. Les algorithmes de comparaison choisis sont K-Moyennes et SingleLinkage appliqués sur les prototypes de la carte SOM après apprentissage. L'indice de Davies-Bouldin [35] est utilisé pour déterminer le meilleur découpage des arbres (SingleLinkage) ou le nombre optimal K de centroïdes pour K-Moyennes.

Pour la classification hiérarchique *SingleLinkage*, la proximité entre deux groupes est définie comme la distance minimum entre deux objets appartenant chacun à un groupe. La méthode *SingleLinkage* est efficace pour la détection de groupes non-elliptiques, mais est sensible au bruit et aux *outliers*.

La qualité de la segmentation à été évaluée à partir d'indices externes (indices de Rand et Jaccard) fréquemment utilisés [63, 64]. Le concept de stabilité est aussi utilisé pour estimer la validité de la segmentation. Pour évaluer la stabilité des différents algorithmes, nous utilisons une méthode de sous-échantillonnage [12].

2.3.2.2 Résultats

Les résultats pour les indices externes montrent que pour ces données, DS2L-SOM est capable de retrouver sans aucune erreur la segmentation attendue et le bon nombre de groupes. Ce n'est pas le cas des autres algorithmes, en particulier lorsque les groupes sont de formes arbitraires, lorsqu'il n'y a pas de structure dans les données ou lorsque les groupes sont en contact (voir Figure 2.14, Tables 2.4 et 2.5). Nous obtenons des résultats similaires à l'algorithme SOM : U*C pour lequel on été crées les données du FCPS (voir [149]), mais en utilisant en moyenne 50 fois moins de neurones ce qui permet une exécution beaucoup plus rapide. De plus, notre algorithme est capable de gérer les problèmes de déchirement grâce aux valeurs associées aux connexions, comme nous l'avons montré dans la section 2.2.2.3.

En ce qui concerne la stabilité (Figure 2.15 et Table 2.6), l'algorithme DS2L-SOM, à l'instar de l'algorithme S2L-SOM, montre d'excellents résultats pour les données regroupées en hypersphères, quelle que soit la dimension (“ Hepta”et “ HighDim”), mais aussi dans les cas où les groupes sont de formes arbitraires en deux dimensions

TABLE 2.4 – Nombre de groupes obtenus avec différentes méthodes de classificassion (SL=SingleLinkage, KM=K-means, SSL=SOM+SingleLinkage, SKM=SOM+K-means).

Données	SL	KM	SSL	SKM	S2L-SOM	DS2L-SOM	**Correct**
HighDim	9	8	4	9	9	9	**9**
Chainlink	15	13	2	11	2	2	**2**
Atom	15	10	6	6	2	2	**2**
Twodiamonds	5	2	2	2	2	2	**2**
Rings	12	15	7	13	3	3	**3**
Spirals	14	10	12	10	3	3	**3**
Hepta	7	7	5	7	7	7	**7**
Random	11	15	10	15	1	1	**1**
Wingnut	17	18	6	11	1	2	**2**
Engytime	18	15	10	10	1	2	**2**

TABLE 2.5 – Comparaison de la qualité de la segmentation avec l'indice de Jaccard moyen (arrondi au centième) selon les bases de données et les méthodes utilisées.

Données	SOM+CAH	SOM+K-Moyenne	S2L-SOM	**DS2L-SOM**
HighDim	43%	100%	100%	**100%**
Spirals	44%	12%	100%	**100%**
Rings	47%	32%	100%	**100%**
Chainlink	50%	17%	100%	**100%**
Atom	44%	57%	100%	**100%**
Hepta	76%	100%	100%	**100%**
Random	88%	07%	100%	**100%**
Twodiamonds	100%	100%	89%	**100%**
Wingnut	90%	59%	50%	**90%**
Engytime	47%	18%	50%	**90%**
Moyenne	63%	50%	89%	**98%**

("Rings" et "Spirals") et lorsque les données ne sont pas structurées ("Random"). On remarque que, dans ce dernier cas, la segmentation obtenue par les méthodes classiques est extrêmement instable. La bonne stabilité des algorithmes S2L-SOM et DS2L-SOM ne sont pas surprenantes. En effet, plusieurs études thèoriques ont montrés que des algorithmes de clustering définissant des frontières de clusters dans des régions peu denses de l'espace tendent à être stables [10, 135, 156], or les algorithmes S2L-SOM et DS2L-SOM cherchent justement à définir les frontières des clusters dans des régions de faible densité.

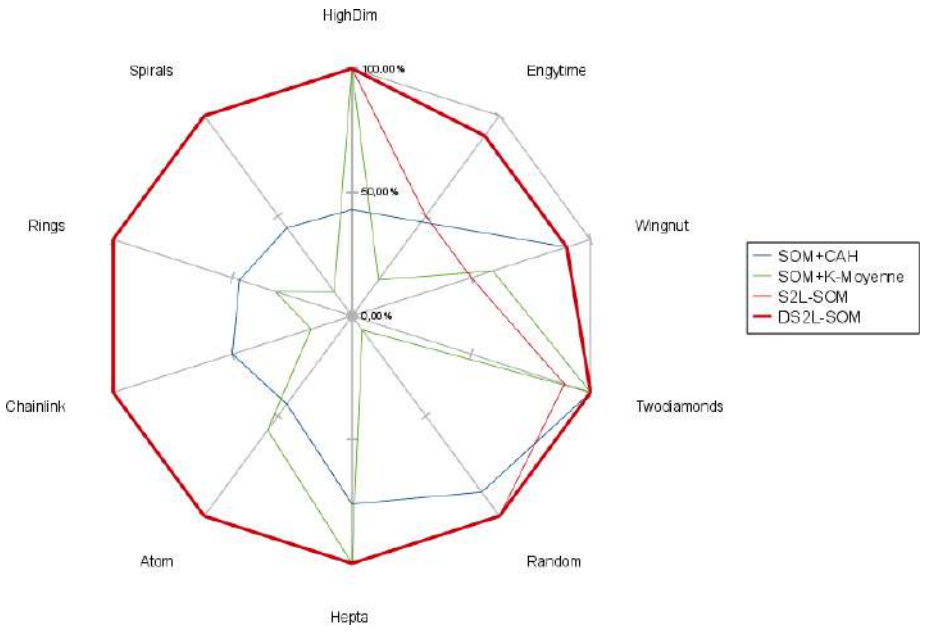


FIGURE 2.14 – Valeur de la qualité de la segmentation selon l'indice de Jaccard pour chaque algorithme sur chaque base de données.

Lorsque les données ne sont pas linéairement séparables dans des dimensions supérieures à deux ("Atom" et "Chainlink"), DS2L-SOM, comme S2L-SOM, est limité par la contrainte topologique en deux dimensions de la carte auto-organisatrice et la stabilité de la segmentation n'est pas maximale. On peut cependant noter que même dans ce cas DS2L-SOM reste plus stable que les autres méthodes. Par ailleurs, lorsque les groupes ne sont définis que par la densité (" Twodiamonds", " Engytime", " Wingnut"), le sous-échantillonnage peut atténuer les fluctuations de densité dans les données, et donc la détection des zones de séparation entre les groupes, ce qui peut diminuer la stabilité de la segmentation. Dans ce cas S2L-SOM est plus stable que DS2L-SOM, parce qu'il ne peut pas séparer ce type de groupes et ne détecte donc qu'un seul groupe pour tous les sous-échantillons.

La visualisation des groupes obtenus confirme ces résultats. En effet, l'algorithme DS2L-SOM est un puissant outil pour visualiser les groupes en deux dimensions.

Les Figures 2.16 et 2.17 présentent deux exemples de résultats obtenus dans cette étude. Dans ces figures, chaque hexagone représente un prototype de la SOM et ses données associées. Les hexagones qui partagent une même couleur sont dans le même groupe. Les hexagones blancs ne font partie d'aucun groupe. Les groupes sont donc aisément et clairement identifiables, ainsi que les zones sans données (unités non connec-

TABLE 2.6 – Comparaison de la stabilité de la classification selon les bases de données et les méthodes utilisées.

Données	SOM+CAH	SOM+K-Moyenne	S2L-SOM	**DS2L-SOM**
HighDim	56%	90%	100%	**100%**
Spirals	99%	46%	97%	**97%**
Rings	90%	66%	100%	**100%**
Chainlink	81%	50%	82%	**82%**
Atom	80%	85%	87%	**87%**
Hepta	71%	98%	99%	**100%**
Random	65%	13%	100%	**100%**
Twodiamonds	100%	99%	84%	**95%**
Wingnut	51%	45%	100%	**80%**
Engytime	90%	38%	100%	**90%**
Moyenne	78%	63%	95%	**93%**

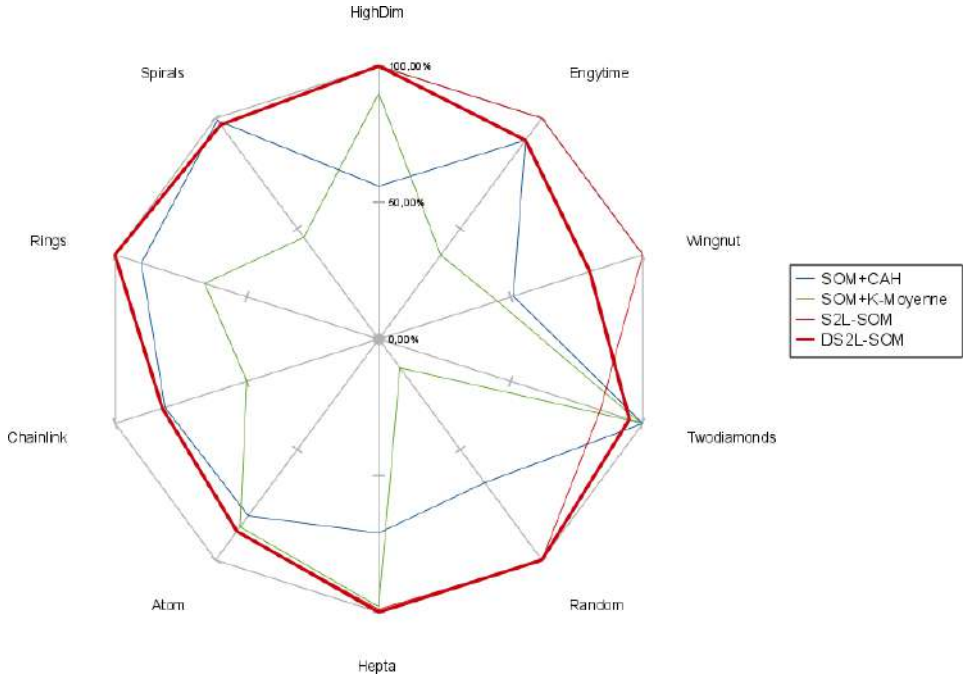
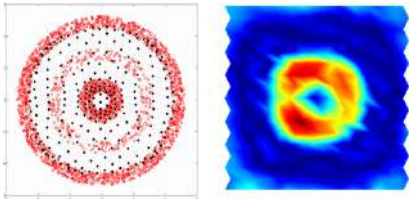
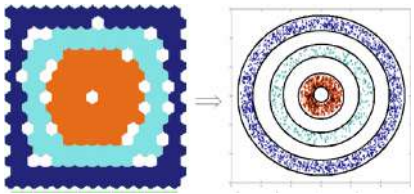


FIGURE 2.15 – Valeur de l'indice de stabilité de la segmentation pour chaque algorithme sur chaque base de données.

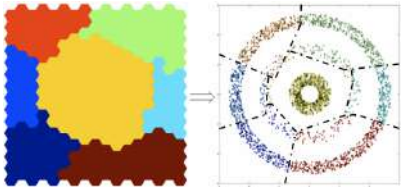
tées). Tel qu'on peut le voir sur les figure 2.16 à 2.17, les résultats obtenus avec l'algorithme DS2L-SOM sont plus proches de la réalité que ceux obtenus avec d'autres méthodes. La figure 2.17 montre que DS2L-SOM est capable de détecter même les



(a) Données d'apprentissage et densité estimée.



(b) Segmentation de la SOM avec DS2L-SOM et projection des étiquettes sur les données.



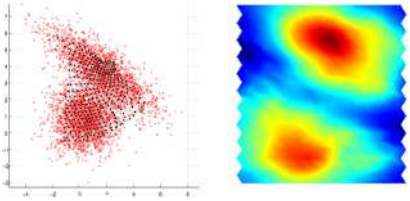
(c) Segmentation de la SOM avec une CAH et projection des étiquettes sur les données.

FIGURE 2.16 – Résultats obtenus à partir d'une CAH ou de DS2L-SOM sur les données "Ring".

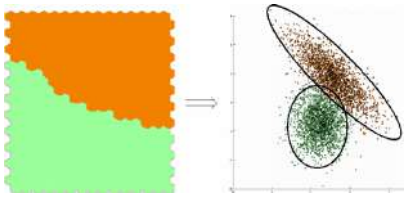
groupes définis par la densité.

2.3.3 Comparaison avec des méthodes récentes

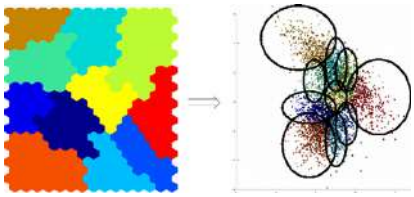
De nombreuses nouvelles méthodes de classification sont proposées régulièrement. Certains algorithmes proposés récemment sont réputés plus performants que les méthodes classiques telles que K-moyennes ou CAH. Dans cette section nous allons mettre en évidence certaines limites des algorithmes récents les plus connus dans le domaine en nous basant sur le travail de [81] qui ont comparé leur propre algorithme, CHAMELEON, aux algorithmes CURE [62] et DBSCAN [44]. Les quatre bases de données utilisées pour la comparaison sont des bases en deux dimensions légèrement bruitées avec des groupes de formes très variées, parfois non linéairement séparables. Nous avons



(a) Données d'apprentissage et densité estimée.



(b) Segmentation de la SOM avec DS2L-SOM et projection des étiquettes sur les données.

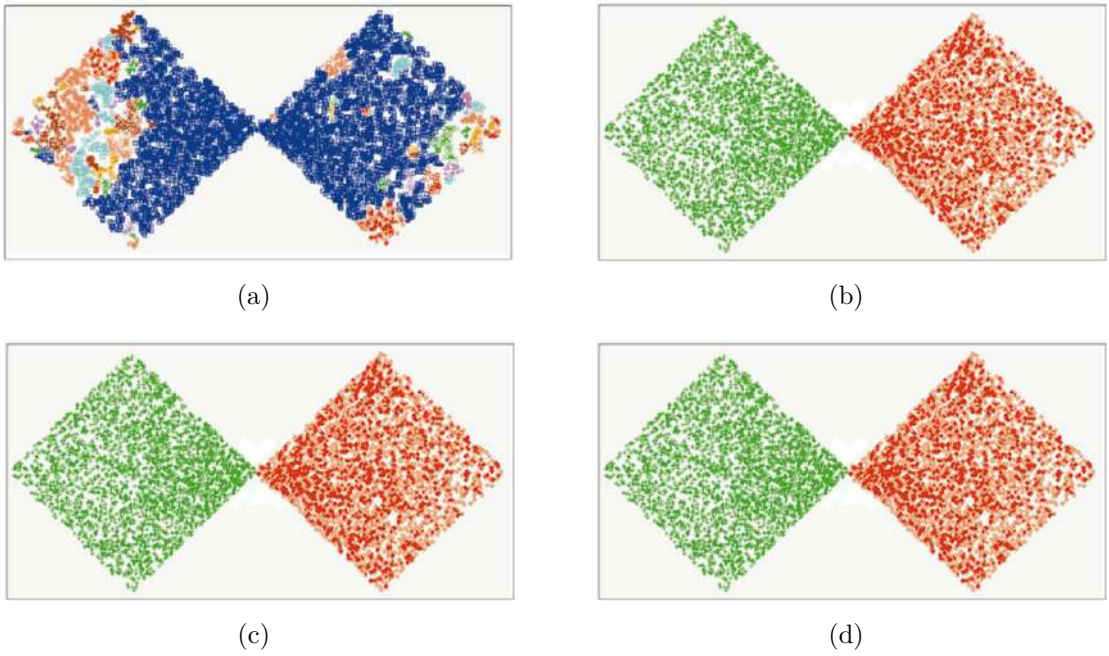


(c) Segmentation de la SOM avec une CAH et projection des étiquettes sur les données.

FIGURE 2.17 – Résultats obtenus à partir d'une CAH ou de DS2L-SOM sur les données "Engytime".

appliqué, sur les mêmes bases de données, DS2L-SOM ainsi qu'un autre algorithme bien connu : l'Analyse Spectrale [137] afin de compléter la comparaison. Les résultats sont présentés par les figures 2.18 à 2.21. Les résultats montrés sont les meilleurs obtenus en faisant varier les différents paramètres des algorithmes, au sens de la pureté des groupes obtenus, c'est à dire en maximisant le nombre de paires de données correctement classées dans un même groupe (selon les regroupements attendus pour ces données).

Ces résultats montrent que, contrairement à DS2L-SOM et CHAMELEON, les algorithmes DBSCAN, CURE et l'Analyse Spectrale ont des difficultés à obtenir une segmentation pertinente sur ces bases de données. En particulier, CURE peine à séparer des groupes non linéairement séparables et DBSCAN a du mal à traiter des groupes de densités différentes ou en contact. Les performances de DS2L-SOM sont aussi bonnes que celles de CHAMELEON pour ces données. La seule différence no-



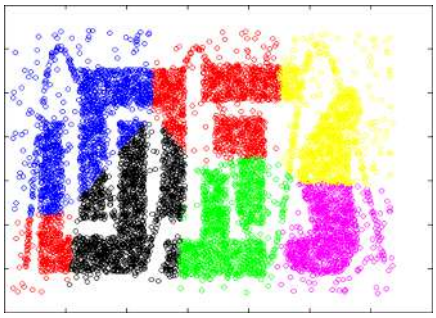
(a) (b)

(c) (d)

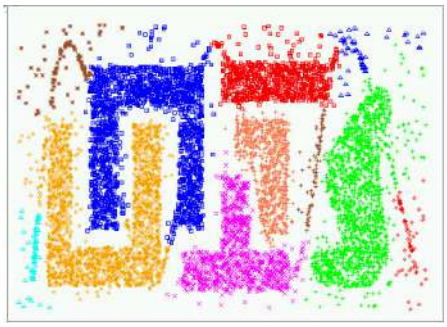
FIGURE 2.18 – Résultats obtenus par (a) DBSCAN, (b) Analyse Spectrale, (c) CHAMELEON et (d) DS2L-SOM sur les données "TwoDiamonds".

table est que DS2L-SOM sépare la branche droite du "Y" inversé de la base "T8". Visuellement cette séparation peut sembler incorrecte vis-à-vis des données. Pourtant, si on visualise une estimation de la densité des données de la base (Figure 2.22(b)) on remarque une zone de très faible densité, peu visible à l'œil nu sur la figure 2.21, séparant deux zones de fortes densités. Les nombreux prototypes de DS2L-SOM ayant servi à l'estimation de la densité garantissent la précision de cette estimation (voir Figure 2.22(a)). Selon nos critères de séparation des groupes de notre algorithme cette zone de faible densité doit être une frontière entre deux groupes. On peut noter que cette séparation est aussi détectée par CURE, en plus de nombreuses autres séparation moins pertinentes.

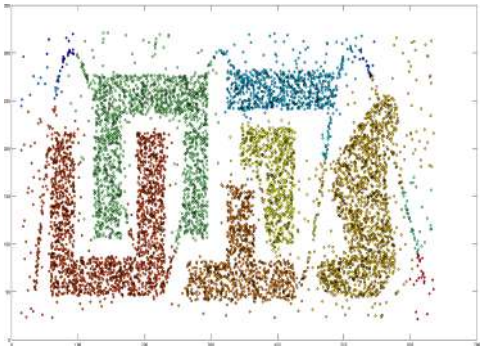
Ces résultats montrent que DS2L-SOM et CHAMELEON sont équivalents en terme de performance sur ces données. Cependant on peut noter que CHAMELEON est basé sur une recherche des K plus proches voisins, tâche de complexité quadratique pour des données en grandes dimensions, alors que la complexité de DS2L-SOM est linéaire pour le nombre de données à analyser. De plus DS2L-SOM permet une visualisation des résultats en deux dimensions, alors que CHAMLEON construit un graphe d-dimensionnel qui, comme avec Neural Gas, ne peut être aisément visualisé.



(a)



(b)

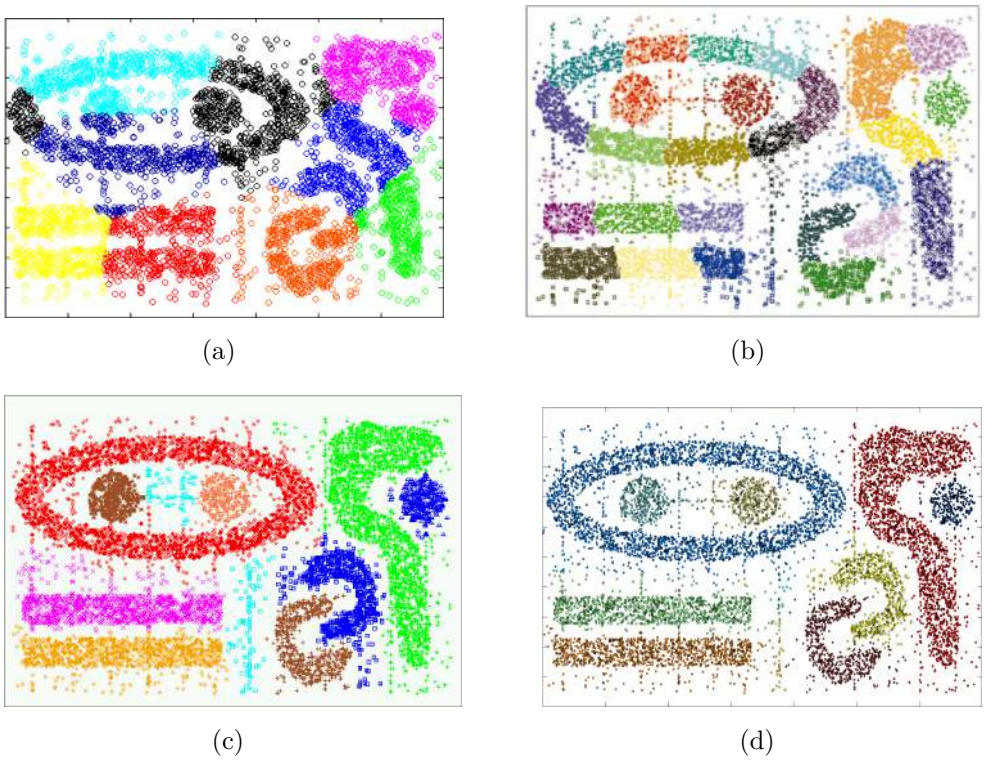


(c)

FIGURE 2.19 – Résultats obtenus par (a) Analyse Spectrale, (b) CHAMELEON et (c) DS2L-SOM sur les données “T4”.

2.3.4 Conclusions

Nous avons proposé daans cette deuxième partie du chapitre une méthode de classification non supervisée qui effectue en même temps une réduction de dimension, par apprentissage d'une SOM, et une classification optimisée utilisant à la fois des informations sur les distances et la densité des données. La densité est estimée à partir de la détection de modes de densité. L'algorithme DS2L-SOM proposé détecte ainsi les régions de haute densité, caractéristiques des regroupements de données similaires, séparés par des régions de faible densité. Les performances de cette méthode ont été évaluées à partir de tests sur une série de problèmes fondamentaux pour la classification, et comparées aux méthodes à deux niveaux classiques, s'appuyant sur CAH ou K-Moyennes. Une comparaison a aussi été effectuée avec des algorithmes de classification plus récents couramment cités. Les résultats expérimentaux démontrent que l'algorithme proposé produit une classification de meilleure qualité que les approches antérieures. Ils montrent aussi que le grand avantage de l'algorithme DS2L-SOM est sa capacité à identifier des groupes de formes non convexes et même des groupes qui se superposent en partie. De plus, dans notre approche, le nombre de groupes est déterminé



(a) (b)

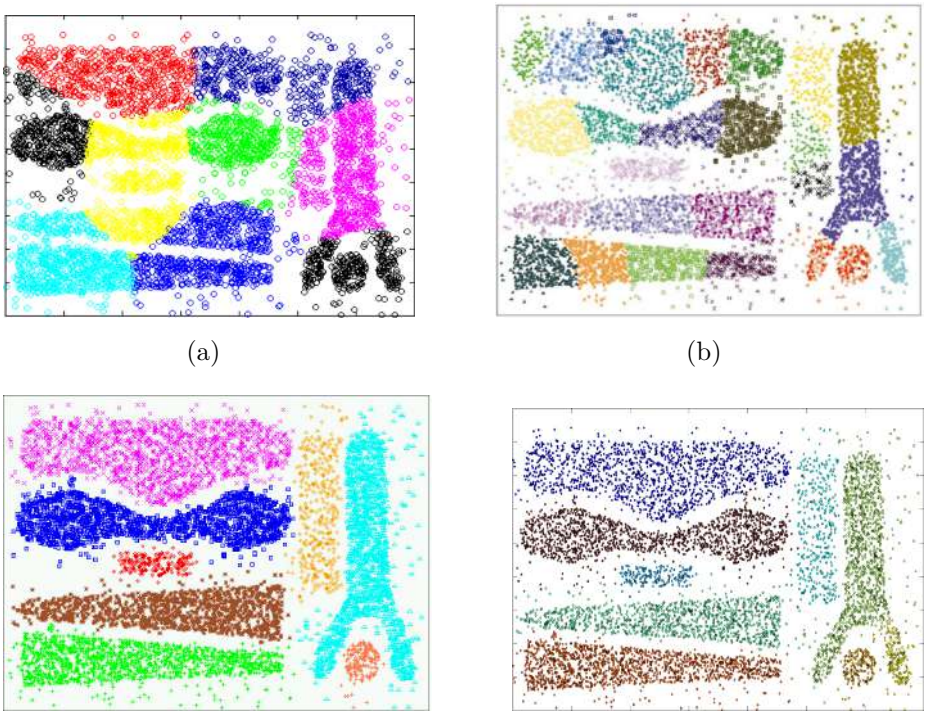
(c) (d)

FIGURE 2.20 – Résultats obtenus par (a) Analyse Spectrale, (b) CURE, (c) CHAMELEON et (d) DS2L-SOM sur les données “T7”.

automatiquement pendant l’apprentissage, c’est-à-dire qu’aucun a priori sur ce nombre n’est requis. Pour finir, la complexité de notre algorithme reste linéaire comparée aux autres approches.

2.4 Conclusion

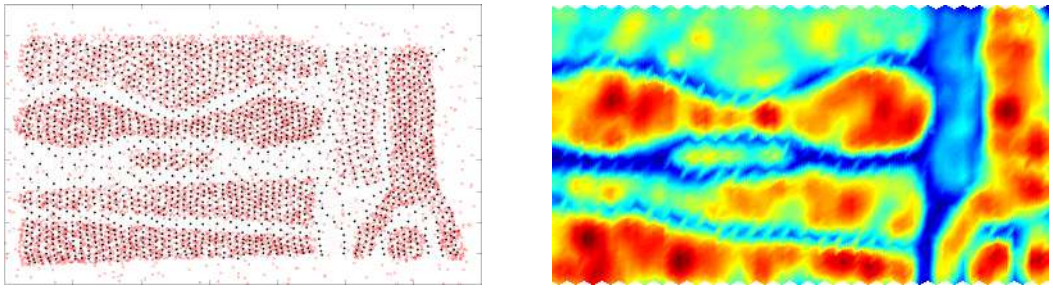
Dans ce chapitre nous avons proposé deux méthodes de classification à deux niveaux simultanés (S2L-SOM et DS2L-SOM). Nous avons utilisé pour cela l’apprentissage d’une carte auto-organisatrice permettant une quantification des données, couplée à une classification efficace de ces données. Des tests sur des données fictives et réelles ont montré que les performances de ces algorithmes sont meilleures que celles des méthodes classiques en temps de calcul et en qualité de la classification. En particulier, ils sont particulièrement efficaces dans la détection de groupes de forme quelconque non linéairement séparables. De plus, le nombre de clusters est détecté automatiquement. L’algorithme DS2L-SOM est aussi particulièrement adapté à la détection de clusters bruités ou en contact. Les puissants outils de visualisation associés aux cartes, du fait



(a) (b)

(c) (d)

FIGURE 2.21 – Résultats obtenus par (a) Analyse Spectrale, (b) CURE, (c) CHAMELEON et (d) DS2L-SOM sur les données “T8”.



(a) (b)

FIGURE 2.22 – Visualisation (a) des prototypes de DS2L-SOM et (b) de la densité estimée sur les données “T8”.

de leur topologie en deux dimensions, permettent d’exploiter au mieux les informations disponibles après apprentissage.

Dans le chapitre suivant nous allons présenter deux extensions des principes mis en œuvre dans ces algorithmes. La première utilise les informations de connectivité pour une amélioration de la qualité de la quantification des données par les prototypes. La

deuxième est une adaptation de S2L-SOM pour la classification de données non vectorielles : les données intervalles. D'autres adaptations sont aussi proposées en Annexes.

Chapitre 3

Relachement de contraintes topologiques et adaptation à d'autres types de données

3.1 Introduction

Les algorithmes présentés dans le chapitre précédent ont de nombreuses qualités. Cependant, deux principales remarques peuvent être formulées :

Premièrement, les méthodes à deux niveaux ont une double tâche à effectuer : la quantification des données et leur partitionnement. Or, les valeurs estimées pendant l'apprentissage de la carte sont des informations utiles pour le partitionnement final des prototypes (et des données), mais pourraient être utilisées aussi pour améliorer la qualité de la quantification. C'est ce que nous proposons dans la première partie de ce chapitre, en utilisant des informations de connectivité apprises selon les mêmes principes que ceux utilisés dans S2L-SOM.

Deuxièmement, les méthodes proposées jusqu'à maintenant ne sont adaptées qu'à l'analyse de données présentées sous forme de vecteurs. Or, de nombreuses autres représentations sont fréquemment utilisées dans la pratique. Nous montrons dans la deuxième partie de ce chapitre qu'il est relativement facile d'adapter ces algorithmes à des données non-vectorielles, en présentant une adaptation de S2L-SOM aux données intervalles.

3.2 Relachement de contraintes topologiques

Nous rappelons qu'une carte Auto-Organisatrice se compose d'un ensemble de neurones artificiels, qui représentent la structure des données. Les neurones sont connectés avec des connexions topologiques pour former une grille à deux dimensions. Deux neurones connectés devraient représenter le même type de données, deux neurones distants (sur la carte) doivent représenter des données différentes. Ces propriétés sont assurées pendant le processus d'apprentissage grâces aux informations de voisinage qui imposent des contraintes topologiques.

Toutefois, dans l'algorithme SOM, l'information topologique est fixée avant le processus d'apprentissage et peut ne pas être pertinente par rapport à la structure des données. Pour résoudre ce problème, certains travaux ont été réalisés afin d'adapter le nombre de neurones au cours du processus d'apprentissage en fonction des données à analyser [55]. Les résultats ont montré que la qualité du modèle est améliorée lorsque le nombre de neurones est appris à partir des données.

En dépit de ces résultats, il y a très peu de travaux qui abordent le problème de l'apprentissage des contraintes topologiques en fonction de la structure des données.

Pourtant, à la fin du processus d'apprentissage, des neurones "voisins" peuvent ne pas représenter des données similaires [103]. Dans l'algorithme False Neighbor-SOM (FN-SOM, [103]), les auteurs proposent de conserver la topologie bidimensionnelle de la SOM, mais en associant à chaque "ligne" ou "colonne" de connexions un indice de voisinage qui est lié à la validité globale de la "ligne" ou de la "colonne" (dans cet algorithme, chaque neurone est limité à seulement quatre voisins ; cf. Figure 3.1). Comme la même valeur est utilisée pour toutes les connexions appartenant à la même "ligne" ou "colonne", la méthode peut conduire à une structure de topologie incorrecte, en particulier pour les bases de données de grande dimension.

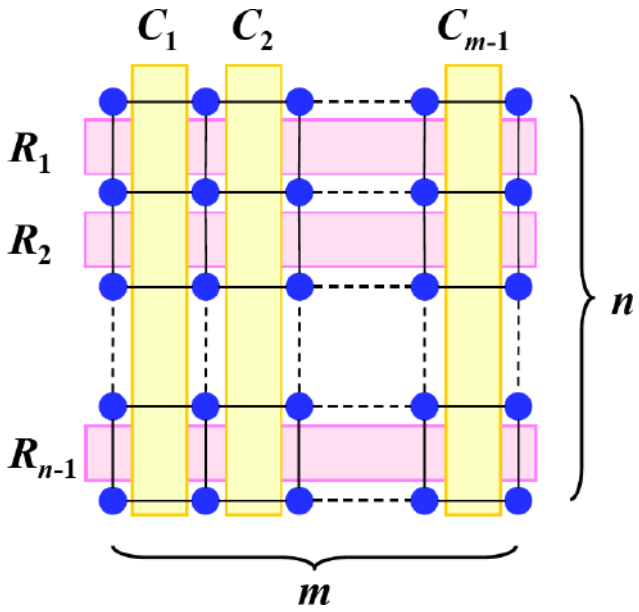


FIGURE 3.1 – Dans FN-SOM des valeurs sont associées à chaque ligne et colonne d'une SOM de topologie rectangulaire.

Nous proposons donc dans cette partie d'améliorer les performances SOM avec un nouvel algorithme, Data-Driven Relaxation – SOM (DDR-SOM), capable d'apprendre la topologie de la carte à partir de la structure de données. Cet algorithme permet une visualisation en deux dimensions des résultats et le nombre de voisins n'est pas limité (un voisinage de six est le plus couramment utilisé dans SOM). L'idée principale est d'utiliser des valeurs associées aux connexions comme dans S2L-SOM, afin de réduire certaines contraintes topologiques entre les neurones qui représentent des données différentes. Ces relâchements de contraintes sont censés améliorer la qualité de la SOM, notamment en réduisant l'erreur de quantification de la carte et l'augmentation du nombre de neurones qui participent réellement à la représentation des données.

3.2.1 Notion de contraintes topologiques dans les Cartes Auto-Organisatrices

L'algorithme DDR-SOM que nous allons présenter dans ce chapitre se base sur la version batch de SOM. Cette version se présente de la façon suivante :

1) **Phase d'initialisation** :
   - Choisir la topologie de la SOM.
   - Initialiser des prototypes w de la carte.
2) **Phase de compétition** :
   - Déterminer le premier BMU u^* pour chaque donnée d'entrée $x^{(k)}$:

$$u^*(x^{(k)}) = \underset{1 \leq i \leq M}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

3) **Phase d'adaptation** :
   - Mettre à jour la fonction Noyau K_{ij} pour chaque pair de neurones en fonction de la température $\lambda(t)$.
   - Mettre à jour les prototypes w_i de la carte pour chaque neurone i de façon à minimiser la fonction de coût :

$$w_i = \frac{\sum_{k=1}^{N} K_{iu^*(x^{(k)})}.x^{(k)}}{\sum_{k=1}^{N} K_{iu^*(x^{(k)})}}$$

   avec la fonction Noyau K_{ij} qui représente les contraintes topologiques à respecter :

$$K_{ij} = \frac{1}{\lambda(t)} \times e^{-\frac{d_M^2(i,j)}{\lambda^2(t)}}$$

   $d_M(i,j)$ est la distance de voisinage entre deux neurones i et j de la SOM, calculée par la distance de Manhattan. Il s'agit du nombre minimal de connexions topologiques entre i et j (voir la Figure 3.2 pour un exemple).
4) **Répéter les phases 2 et 3** jusqu'à ce que $t = t_{max}$

3.2.2 Relâchement des contraintes topologiques guidé par les données

3.2.2.1 Principe

Dans l'algorithme DDR-SOM, comme dans S2L-SOM, nous proposons d'associer à chaque connexion de voisinage une valeur réelle v qui indique la pertinence des neurones

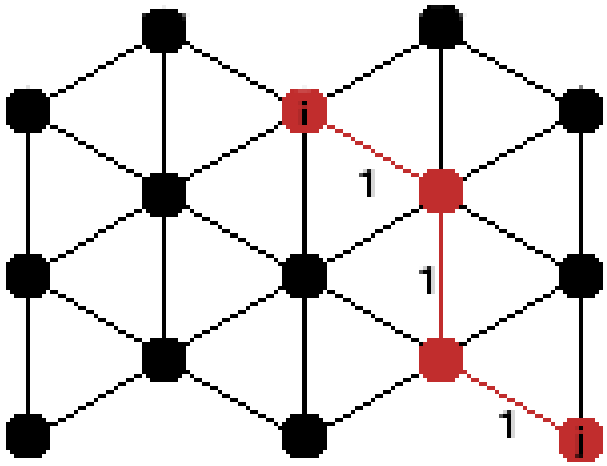


FIGURE 3.2 – Distance de Manhattan pondérée entre les neurones i et j pour une topologie hexagonale. Ici $d_M(i,j) = 3$, il s'agit du nombre minimal de connexions topologiques entre i et j.

connectés. Compte tenu de la contrainte d'organisation de la SOM, les deux neurones les plus représentatifs d'un ensemble de données doivent être reliés par une liaison de voisinage. Ainsi, une paire de neurones voisins, qui sont tout deux de bon représentant d'un même ensemble de données devraient être fortement connectés, tandis qu'une paire de neurones voisins qui ne représentent pas le même type de données doit être faiblement connectés. On associe donc à chaque connexion de voisinage une valeur variant de 1 (lien fort) à 2 (lien faible) en utilisant une fonction logistique qui dépend du nombre de données bien représentées par chacun des deux neurones connectés.

Ces valeurs sont ensuite utilisées pour estimer une distance de Manhattan pondérée $d_{WM}(i,j)$ entre chaque paire de neurones voisins i et j. Cette distance est le nombre de connexions entre i et j, pondéré par les valeurs associées à chaque connexion (voir la Figure 3.3 pour un exemple).

Nous utilisons l'algorithme de Johnson [79] pour calculer :
- le chemin le plus court le long des connexions de voisinages entre chaque paire de neurones.
- la longueur de ce chemin en fonction des valeurs v associées à chaque connexion.

De cette manière, la distance entre deux neurones est supposée refléter le voisinage "réel" de ces deux neurones.

Les valeurs de connexion et les distances pondérées entre neurones sont mises à jour au cours de l'apprentissage. De cette façon, la SOM obtenue devrait être un meilleur représentant de la base de données que l'algorithme SOM classique.

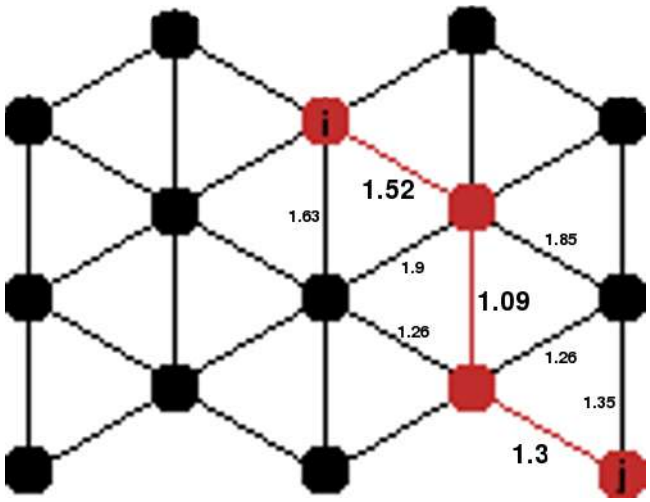


FIGURE 3.3 – Distance de Manhattan pondérée entre les neurones i et j pour une topologie hexagonale. Ici $d_{WM}(i,j) = 1.52 + 1.09 + 1.3 = 3.91$, il s'agit du plus court chemin entre i et j en fonction des valeurs de connexion v.

3.2.2.2 Nouvel Algorithme

L'algorithme DDR-SOM procède en trois phases :

1) **Phase d'initialisation** :
   - Choisir la topologie de la SOM.
   - Initialiser des prototypes w de la carte.
   - Initialiser à zéro toutes les valeurs des connexions de voisinage v.
2) **Phase de compétition** :
   - Déterminer le premier BMU u^* et le second BMU u^{**} pour chaque donnée d'entrée $x^{(k)}$:

$$u^*(x^{(k)}) = \underset{1 \leq i \leq M}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

   et

$$u^{**}(x^{(k)}) = \underset{1 \leq i \leq M,\ \ i \neq u^*}{Argmin} \parallel x^{(k)} - w_i \parallel^2$$

   - Mettre à jour les valeurs v des connexions de voisinage selon la règle suivante :

$$v_{i,j} = \frac{1 + 2e^{\frac{Nth - N(i,j)}{\sigma Nth}}}{1 + e^{\frac{Nth - N(i,j)}{\sigma Nth}}}$$

   Avec $v_{i,j}$ une fonction logistique variant entre 1 (lorsque i et j représentent les mêmes données) et 2 (lorsque i et j représentent des données différentes).

$N(i,j)$ est le nombre de données (x) ayant i et j dans $\{u^*(x), u^{**}(x)\}$. Nth est la valeur théorique de $N(i,j)$ sous l'hypothèse d'homogénéité, autrement dit Nth est la moyenne de $N(i,j)$ sur toutes les connexions de voisinage.

3) **Phase d'adaptation** :
   - Recalculer la distance pondérée $d_{WM}(i,j)$ pour chaque paire de neurones i et j en fonction des valeurs v associées aux connexions de voisinage.
   - Recalculer la fonction Noyau K en fonction de la température $\lambda(t)$ et de la distance $d_{WM}(i,j)$.
   - Mettre à jour les prototypes w_i de la carte pour chaque neurone i de façon à minimiser la fonction de coût :

$$w_i = \frac{\sum_{k=1}^{N} K_{iu^*(x^{(k)})}.x^{(k)}}{\sum_{k=1}^{N} K_{iu^*(x^{(k)})}}$$

4) **Répéter les phases 2 et 3** jusqu'à ce que $t = t_{max}$

3.2.3 Résultats expérimentaux

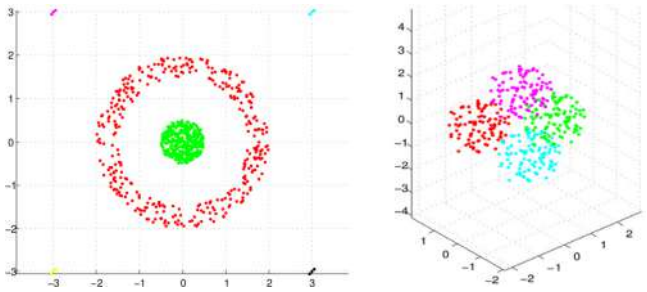
3.2.3.1 Description des bases de données utilisées

Pour tester la validité du nouvel algorithme, nous avons utilisé 10 bases de données artificielles et réelles de tailles et de dimensions variables. Les caractéristiques de chaque base sont décrites dans la Table 3.1.

TABLE 3.1 – Description des bases de données utilisées.

Base de données	Type	Taille	Dimension
Target	Artificielle	770	2
TwoDiamonds	Artificielle	800	2
Hepta	Artificielle	212	3
Tetra	Artificielle	400	3
Iris	Réelle	150	4
Harot	Réelle	132	6
Housing	Réelle	506	13
Wine	Réelle	178	13
Cockroach	Réelle	1369	3
Chromato	Réelle	134	60

Les bases de données “Target”, “TwoDiamonds”, “Tetra” et “Hepta” viennent du “Fundamental Clustering Problem Suite” (FCPS) [149]. Il s’agit de données artificielles de faible dimension dont la structure est parfaitement connue. Elles sont souvent utilisées comme bases de test pour les algorithmes de clustering [103, 149].



(a) “Target” et “Tetra”.

FIGURE 3.4 – Visualisation des bases de données.

Les bases de données “Housing”, “Harot”, “Iris” et “Wine” sont des données réelles bien connues de l’UCI repository [49]. Elles sont de tailles et de dimensions variables. Pour finir, “Cockroach” et “Chromato” sont des bases de données réelles très bruitées provenant d’expériences dans le domaine de la biologie.

Ces bases de données représentent en partie la diversité des problèmes pouvant être rencontrés par les utilisateurs de SOM lors de l’analyse de bases de données.

3.2.3.2 Estimation de la qualité d’une SOM

Pour évaluer les performances en apprentissage de notre algorithme, nous utilisons trois indices classiques de qualité pour les algorithmes de type SOM :

Erreur de Quantification (Qe) :

Cet indice mesure la distance moyenne entre chaque donnée et son BMU [87].

$$Qe = \frac{1}{N} \sum_{k=1}^{N} \parallel x^{(k)} - w_{u^*(x^{(k)})} \parallel^2$$

Plus la valeur de Qe est petite, plus la qualité de l’algorithme est grande.

Erreur Topologique (Te) :

Te décrit la façon dont la SOM préserve la topologie de l'ensemble de données étudiées [85]. Elle mesure la proportion des données ayant les deux premiers BMU non adjacents (non reliés par une connexion de voisinage). Une faible valeur de Te est signe de qualité. Contrairement à l'erreur de Quantification, Te prend en compte la structure de la SOM.

Utilisation des Neurones (Ne) :

Ne mesure le pourcentage de neurones qui ne sont jamais le BMU d'une donnée de la base [28]. Une bonne SOM doit avoir une faible Ne, c'est-à-dire que chaque neurone devrait être utilisé dans la représentation des données.

Dans toutes les expériences suivantes, les indices sont normalisés afin de pouvoir comparer efficacement les résultats sur les différentes bases de données. Pour représenter un gain ou une perte par rapport à l'algorithme classique de SOM, chaque indice est divisé par la valeur obtenue avec l'algorithme SOM (l'erreur de SOM est donc toujours égale à 1). Pour toutes les expériences de cette section, nous avons utilisé le paquet Matlab SOM-Toolbox [152], tous les paramètres des SOM ont été fixés aux valeurs par défaut (en particulier, nous utilisons une grille hexagonale comme topologie initiale de la carte).

3.2.3.3 Effet d'un relâchement des contraintes topologiques guidé par l'utilisateur

Le principe essentiel du nouvel algorithme est de réduire la contrainte topologique de la SOM en augmentant la distance entre les neurones. Dans DDR-SOM, ces modifications sont guidées par les données pour optimiser la qualité finale de la SOM.

La première étape de notre expérimentation est d'analyser comment se comporte une SOM si on diminue de façon triviale les contraintes topologiques. Nous prévoyons qu'une plus faible contrainte conduit à un meilleur modèle (c.-à-d. de plus faibles erreurs de Quantification et Utilisation des Neurones), mais conduit également à une augmentation de l'erreur Topologique, puisque la fonction des contraintes topologiques de la SOM est de réduire cette erreur.

Pour le vérifier, nous avons calculé les erreurs de Quantification, d'Utilisation des Neurones et Topologique pour chaque base de données à partir des résultats de différents relâchements des contraintes topologiques de l'algorithme SOM, où chaque distance entre deux neurones est multipliée par une valeur constante (voir Figure 3.5 à 3.7). Plus cette constante est grande, plus les contraintes topologiques sont faibles.

Par exemple, SOM(α) est similaire à l'algorithme SOM, mais $d(i,j) = \alpha \times d_M(i,j)$. Nous avons testé différentes valeurs de cette constante, de SOM(1), similaire à SOM, à SOM(10), où les neurones sont pratiquement indépendants (dans ce cas le comportement de l'algorithme est similaire à celui d'un K-Moyennes).

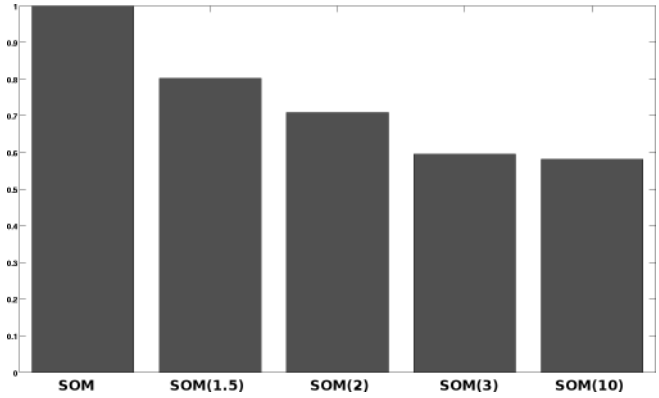


FIGURE 3.5 – Visualisation de la valeur moyenne de Qe sur l'ensemble des bases de données pour différents relâchements des contraintes topologiques dans SOM.

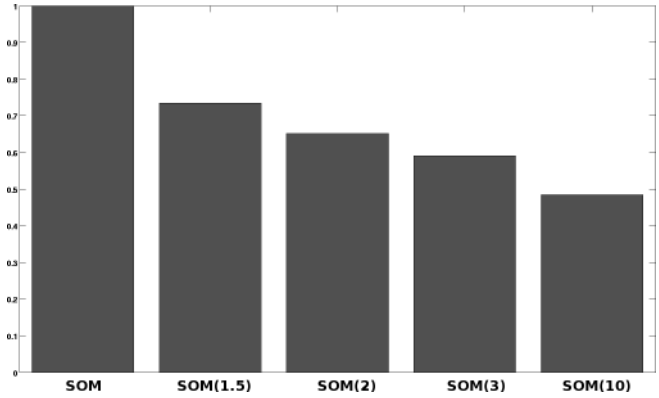


FIGURE 3.6 – Visualisation de la valeur moyenne de Ne sur l'ensemble des bases de données pour différents relâchements des contraintes topologiques dans SOM.

Les résultats sont résumés dans les Figure 3.5 à 3.7. Ces figures montrent la valeur moyenne de Qe, Ne et Te sur toutes les bases de données pour différents relâchements des contraintes topologiques . Comme prévu, Ne et Qe diminuent lorsque les contraintes topologiques s'affaiblissent, tandis que Te augmente fortement.

Puisque le gain en Ne et Qe est associé à une perte en Te, nous proposons de définir une Erreur Générale qui reflète les compromis entre Ne, Qe et Te :

$$Ge = Te^2 \times Ne \times Qe$$

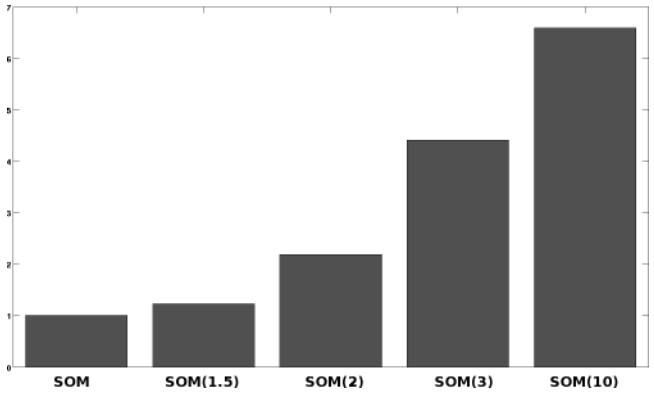


FIGURE 3.7 – Visualisation de la valeur moyenne de Te sur l'ensemble des bases de données pour différents relâchements des contraintes topologiques dans SOM.

Ge est le produit de deux compromis : Ne vs. Te et Qe vs. Te. La valeur de Ge est plus faible lorsque le gain en Ne et Qe est supérieur à la perte en Te par rapport à l'algorithme SOM. Ge est plus grande dans le cas contraire. La Figure 3.8 représente la valeur moyenne de Ge sur toutes les bases de données pour différentes valeurs de α.

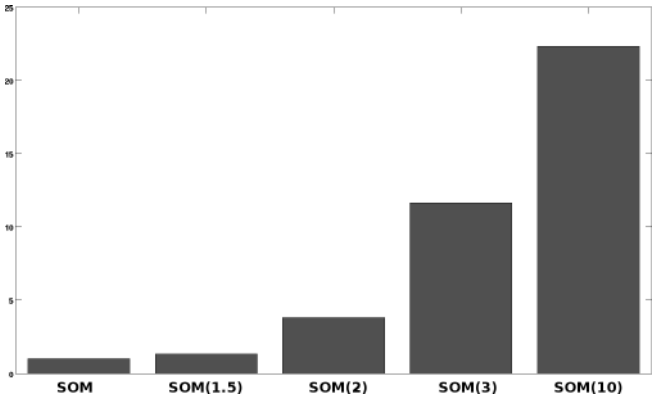


FIGURE 3.8 – Visualisation de la valeur moyenne de Ge sur l'ensemble des bases de données pour différents relâchements des contraintes topologiques dans SOM.

Ces résultats montrent que, sous l'effet d'une diminution des contraintes topologiques guidée par l'utilisateur, le gain en Ne et Qe ne peut pas compenser la perte en Te. Ainsi, le meilleur compromis est d'utiliser l'algorithme SOM classique !

Maintenant la question est : peut-on utiliser les données d'apprentissage pour trouver une relaxation des contraintes topologiques qui soit un meilleur compromis que la SOM ?

3.2.3.4 Évaluation du nouvel algorithme guidé par les données

Pour évaluer la qualité de DDR-SOM, nous avons comparé SOM et différentes versions de DDR-SOM avec différentes valeurs du paramètre σ. Les Figure 3.9 à 3.11 montrent les valeurs moyennes de Qe, Ne et Te. Les valeurs de Ge pour chaque base de données et chaque version de DDR-SOM sont indiquées dans la Table 3.2. La Figure 3.12 montre la valeur moyenne de Ge sur toutes les bases de données. Enfin, la Figure 3.13 illustre la qualité de DDR-SOM en comparaison de l'algorithme SOM.

TABLE 3.2 – Valeurs de Ge obtenues par DDR-SOM pour chaque base de données avec différentes valeurs de σ.

	DDR(1)	DDR(1/2)	DDR(1/5)	DDR(1/10)
Target	0,57	0,39	0,87	0,89
TwoDiamonds	0,22	0,27	0,22	0,11
Hepta	1,82	0,62	0,97	0,57
Tetra	1,17	0,53	1,61	1,12
Iris	0,09	0,21	0,29	0,28
Harot	0,79	0,17	0,06	0,38
Housing	0,83	0,54	0,35	0,55
Wine	0,51	0,14	0,13	0,33
Cockroach	0,62	0,42	0,52	0,54
Chromato	0,12	0,08	0,09	0,09

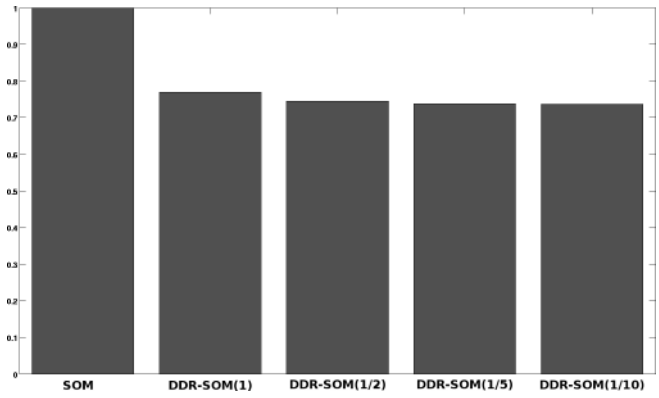


FIGURE 3.9 – Visualisation de la valeur moyenne de Qe sur l'ensemble des bases de données pour différentes valeurs de σ avec DDR-SOM.

Nous pouvons noter que les contraintes topologiques moyennes de DDR-SOM sont similaires à celles de SOM(1.5). Cependant, comme nous pouvons le constater, les performances de l'algorithme DDR-SOM sont bien meilleures que celles de SOM(1.5)

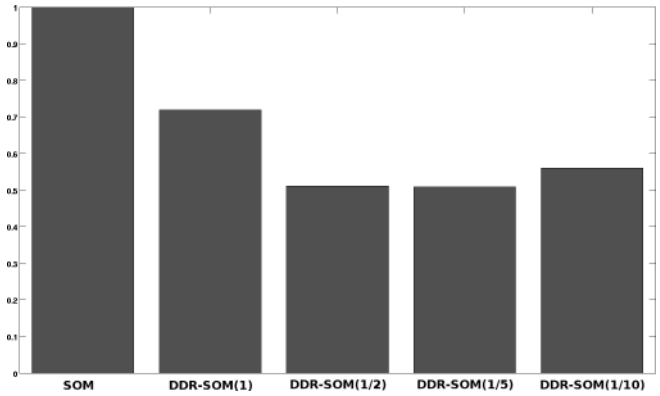


FIGURE 3.10 – Visualisation de la valeur moyenne de Ne sur l'ensemble des bases de données pour différentes valeurs de σ avec DDR-SOM.

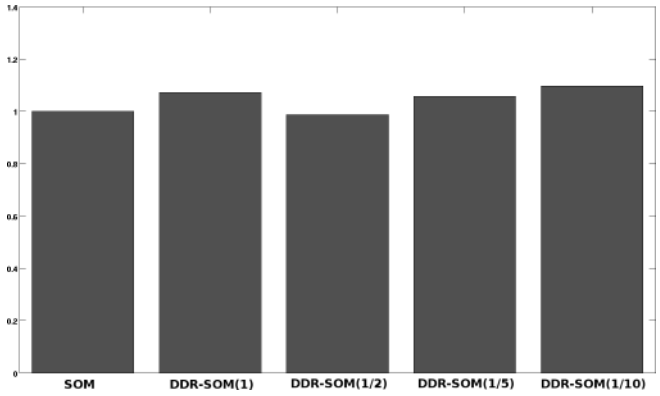


FIGURE 3.11 – Visualisation de la valeur moyenne de Te sur l'ensemble des bases de données pour différentes valeurs de σ avec DDR-SOM.

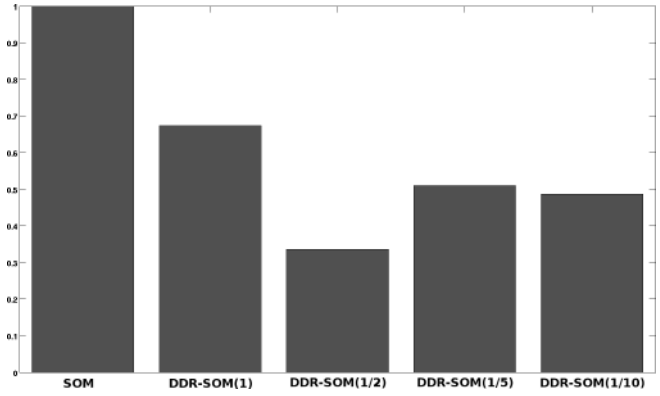
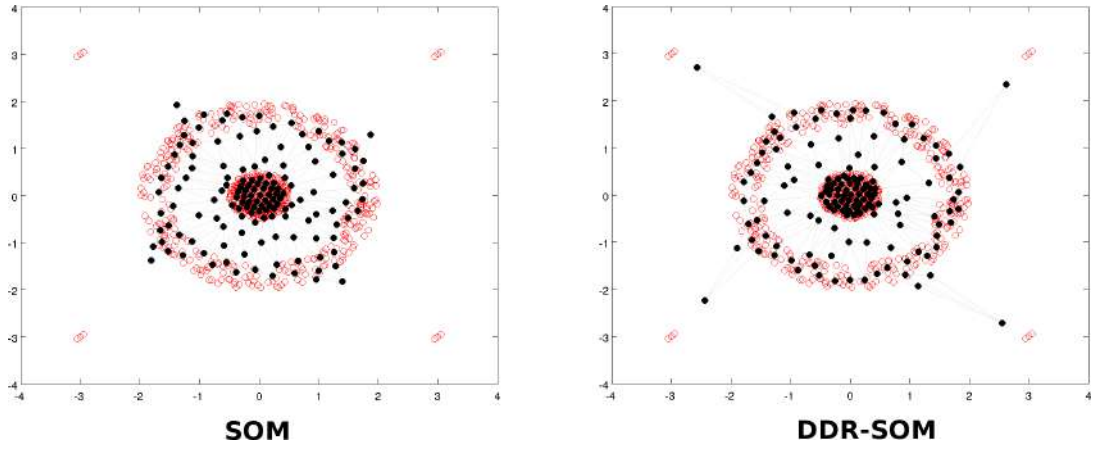
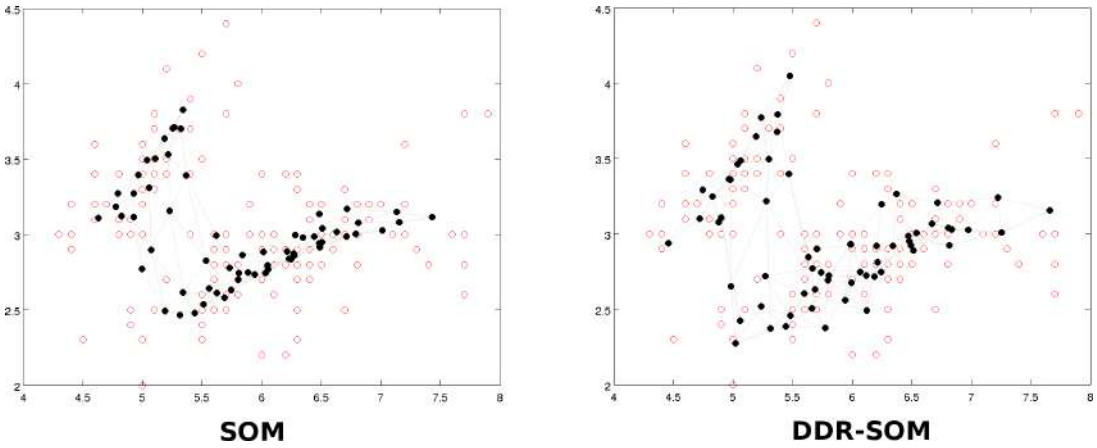


FIGURE 3.12 – Visualisation de la valeur moyenne de Ge sur l'ensemble des bases de données pour différentes valeurs de σ avec DDR-SOM.

et de SOM, autrement dit le gain en Ne et Qe est supérieur à la perte en Te, par rapport à SOM. En effet, avec DDR-SOM, l'erreur de Quantification est semblable à celle de SOM(2) et Ne est très faible, semblable à SOM(10), tandis que l'erreur Topologique de DDR-SOM est semblable à l'erreur obtenue par l'algorithme SOM classique.



(a) Données "Target".



(b) Données "Iris" (2 premières variables).

FIGURE 3.13 – Visualisation des résultats obtenus avec SOM et DDR-SOM(1/2) pour deux bases de données. Les données sont en rouge et la grille de prototypes à la fin de l'apprentissage est représentée en noir.

Nous pouvons faire deux commentaires à partir de ces résultats :

1) La qualité de DDR-SOM est meilleure que celle de SOM pour toutes les valeurs de σ, cependant une valeur de $\sigma = 1/2$ semble donner de meilleurs résultats pour ces bases de données.

2) Le gain en Ge en comparaison de SOM a tendance à être plus élevé pour les bases de données de grandes dimensions (par exemple "Chromato", "Wine", etc ...).

De plus, si l'on se base sur l'article de [103], on montre que, dans le cas de cartes 10x10 à topologie rectangulaire, DDR-SOM est en moyenne 25% plus performant que FN-SOM pour les quelques bases de données utilisées par les auteurs ("Tetra", "Hepta", "Target" et "Iris").

3.2.4 Conclusion

Dans cette partie, nous avons proposé un nouvel algorithme adapté de SOM dans le but d'améliorer la qualité du modèle obtenu par apprentissage. Pour cela nous avons utilisé un relâchement des contraintes topologiques de la SOM guidé par les données à partir d'estimations des valeurs de connexions topologiques (selon le principe de S2L-SOM). Nous avons défini une erreur globale qui représente le compromis entre les erreurs Topologiques, de Quantification et d'Utilisation des Neurones.

Les expériences sur des bases de données artificielles et réelles montrent que l'algorithme DDR-SOM obtient de meilleurs résultats que l'algorithme SOM. Nous avons également montré que cette amélioration n'est pas obtenue avec une relaxation triviale des contraintes topologiques, en raison d'une forte augmentation de l'erreur Topologique. Une diminution des contraintes guidée par les données semble être une bonne solution pour améliorer le compromis $NeQe/Te$ de la SOM.

3.3 Adaptation aux données intervalles

[1] Les algorithmes S2L-SOM, DS2L-SOM et DDR-SOM que nous avons proposés sont conçus pour traiter des données représentées dans un espace vectoriel de $\mathbb{R}^d$. Autrement dit les données sont des vecteurs de d valeurs réelles. Ce type de représentation est très fréquemment utilisé pour analyser des données issues de mesures physiques, de comptages ou d'indices, mais il existe de nombreux autres types d'information qui ne peuvent être décrits par des vecteurs. C'est le cas de données complexes décrites par exemple à l'aide d'un texte, d'une image ou d'une structure hiérarchique ...

Il est possible de modifier les algorithmes de clustering que nous avons proposés précédemment afin de pouvoir analyser des données non vectorielles. Cela implique principalement deux modifications : la définition de prototypes adaptés et la définition d'une mesure de distance entre données et prototypes.

1. Ce travail à été réalisé en collaboration avec M. Renaud Destenay sur proposition du Prof. André Hardy de l'université de Namur en Belgique.

Pour illustrer ces possibilités, nous proposons dans cette partie une adaptation de S2L-SOM pour l'analyse de données intervalles. Les données intervalles sont définies dans un espace vectoriel non pas par un point, mais par un hyper-rectangle. Les intervalles sont souvent employés pour modéliser des quantités qui varient entre deux bornes, inférieure et supérieure, sans faire d'autres hypothèses sur la distribution entre ces bornes [18, 27, 66].

3.3.1 Les données intervalles

3.3.1.1 Définitions

Dans un espace vectoriel, une donnée intervalle x est une donnée définissant un intervalle borné et fermé de $\mathbb{R}^d$ (autrement dit, x est un hyper-rectangle de $\mathbb{R}^d$, cf. Figure 3.14 pour un exemple). Une donnée intervalle peut être définie par deux vecteurs de $\mathbb{R}^d$, les bornes inférieures ($bi = [bi_1, \ldots, bi_d]$) et supérieures $bs = [bs_1, \ldots, bs_d]$, telles que :

$$\forall j \in [1, \ldots, d], bi_j < bs_j$$

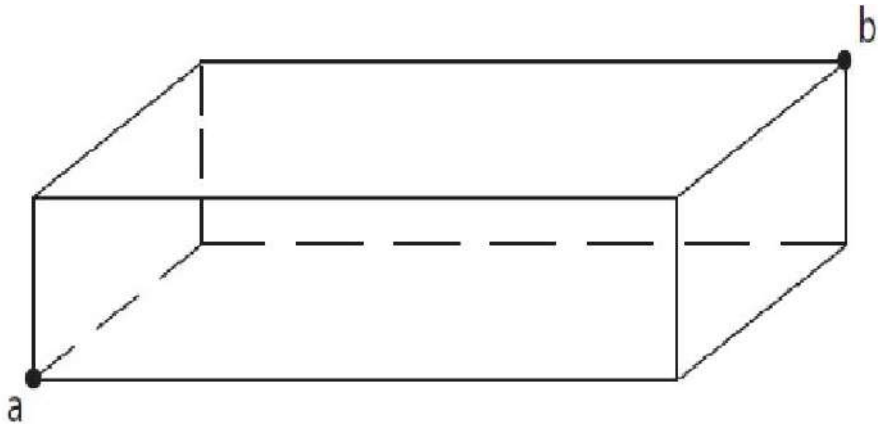


FIGURE 3.14 – Borne inférieure (a) et borne supérieure (b) d'un hyper-rectangle de dimension 3.

3.3.1.2 Les distances

De nombreuses mesures de distances ont été définies pour comparer des intervalles. Pour ce travail, nous en avons choisi trois qui nécessitent peu de puissance de calcul :

la distance au sommet et les distances L1 et L2.

Soit d la dimension de l'espace de représentation des données. La distance au sommet $d_S(x, x')$ est proportionnelle à la somme des distances euclidiennes entre les deux bornes inférieures et les deux bornes supérieures :

$$d_S(x, x') = 2^{d-1} \left(\| \, bi - bi' \, \|^2 + \| \, bs - bs' \, \|^2 \right)$$

La distance L1 se définit de la façon suivante :

$$d_{L1}(x, x') = \sum_{j=1}^{d} max\{|bi_j - bi'_j|, |bs_j - bs'_j|\}$$

La distance L2 est proche de L1, elle s'écrit comme suit :

$$d_{L1}(x, x') = \sqrt{\sum_{j=1}^{d} max\{|bi_j - bi'_j|^2, |bs_j - bs'_j|^2\}}$$

La Figure 3.15 montre un exemple de mesure de distance L1 et L2.

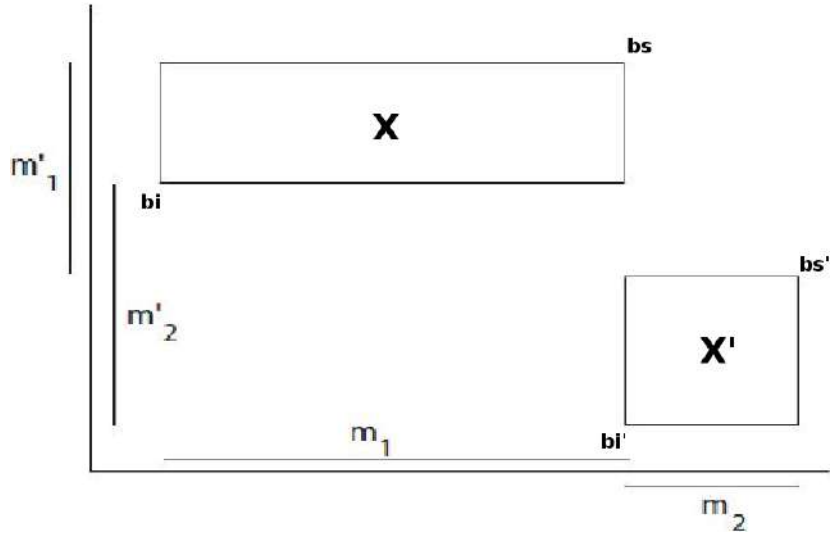


FIGURE 3.15 – Exemple de mesure de distance L1 et L2. Ici $d_{L1}(x, x') = m_1 + m'_1$ et $d_{L2}(x, x') = \sqrt{{m_1}^2 + {m'_1}^2}$.

3.3.2 Adaptation de S2L-SOM aux données intervalles

L'idée principale pour adapter S2L-SOM aux données intervalles est de définir un prototype de la SOM comme une donnée intervalle, c'est à dire un couple de vecteurs : borne inférieure et borne supérieure. Lors de l'apprentissage de la SOM, les deux bornes de chaque prototype vont être mises à jour pour une meilleure représentation des données (voir la Figure 3.16 pour un exemple).

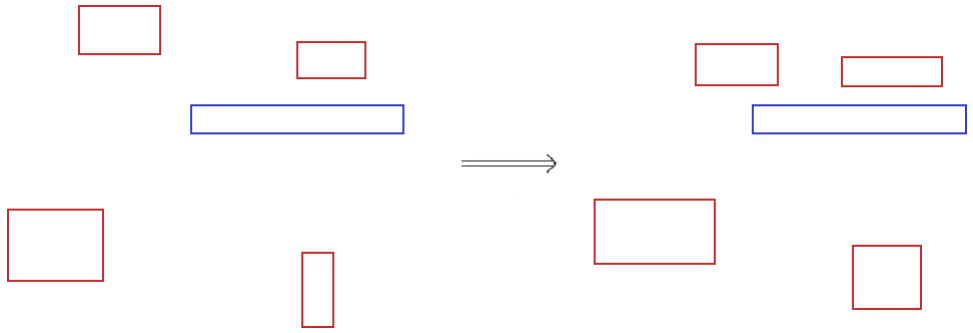


FIGURE 3.16 – Exemple de mise-à-jour des prototypes.

L'algorithme se présente alors de la façon suivante :

Soient $X = \{x^{(i)}\}_{i=1...N}$ un jeu de données intervalles où $x^{(i)} = [bi^{(i)}, bs^{(i)}]$, $d(x, y)$ une distance entre deux intervalles et t_{max} un nombre d'itération maximal.

1) **Phase d'initialisation** :
   - Définir la topologie de la carte.
   - Initialiser aléatoirement tous les prototypes w_j pour chaque neurone j, où $w_j = [bi_j, bs_j]$.
   - Initialiser les connexions ν entre chaque couple de neurones i et j :

$$\forall i, j \quad \nu_{ij} = 0$$

2) **Phase de compétition** :
   - Présenter une donnée $x^{(k)}$ choisie aléatoirement.
   - Parmi les M neurones, choisir les deux meilleurs $u^*(x^{(k)})$ et $u^{**}(x^{(k)})$ pour représenter cette donnée :

$$u^*(x^{(k)}) = \underset{1 \leq i \leq M}{Argmin}\, d(x^{(k)}, w_i)$$
$$u^{**}(x^{(k)}) = \underset{1 \leq i \leq M, \quad i \neq u^*}{Argmin}\, d(x^{(k)}, w_i)$$

– Augmenter la valeur de la connexion entre $u^*(x^{(k)})$ et $u^{**}(x^{(k)})$ et diminuer les valeurs des autres connexions issues de $u^*(x^{(k)})$:

$$\nu_{u^*u^{**}}(t) = \nu_{u^*u^{**}}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*u^{**}}(t-1) - 1\right)$$
$$\nu_{u^*i}(t) = \nu_{u^*i}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*i}(t-1)\right) \quad \forall i \text{ voisin de } u^*$$

Avec :

$$r(t) = \frac{1}{1 + e^{-\left(\frac{t}{t_{max}}\right)}}$$

3) **Phase d'adaptation** :
   – Mettre à jour les bornes des prototypes w_j de chaque neurone j selon les règles :

$$bi_j(t) = bi_j(t-1) - \varepsilon(t)K_{ju^*(x^{(k)})}(bi_j(t-1) - bi^{(k)})$$

$$bs_j(t) = bs_j(t-1) - \varepsilon(t)K_{ju^*(x^{(k)})}(bs_j(t-1) - bs^{(k)})$$

4) **Répéter les phases 2 et 3** jusqu'à ce que les mises à jours des prototypes soient négligeables.
5) **Clustering final** : Déterminer $P = \{C_i\}_{i=1,\dots,L}$, l'ensemble des L groupes de neurones connectés tels que $\nu > 0$. Associer à chaque donnée $x^{(k)}$ la classe du neurone $u^*(x^{(k)})$.

3.3.3 Expériences et Résultats

3.3.3.1 Protocole expérimental

Nous proposons de tester l'efficacité de l'algorithme pour trois types de distances entre données (cf. section 3.3.1.2) et trois types d'initialisation des prototypes.

L'initialisation "données" sélectionne aléatoirement et sans répétition n données et les prend comme prototypes initiaux.

L'initialisation "point" commence par déterminer le plus petit hyper-rectangle enveloppant toutes les données. Puis n points sont sélectionnés aléatoirement dans cet hyper-rectangle. On peut noter qu'un point est un hyper-rectangle dont la borne inférieure et la borne supérieure sont égales.

La dernière initialisation, "lininit", prend aussi pour prototypes initiaux des points. Pour déterminer leurs positions, on effectue une Analyse en Composante Principale

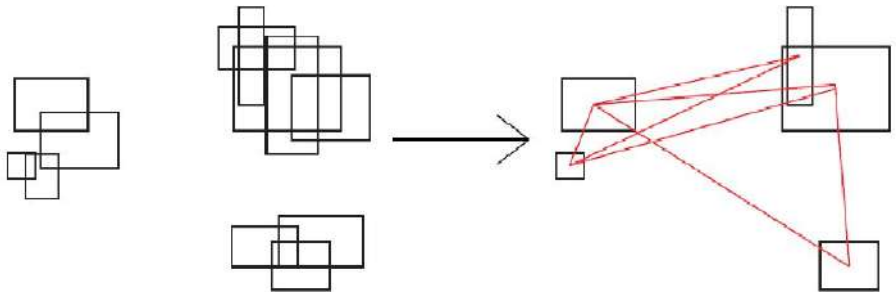


FIGURE 3.17 – Exemple d'initialisation "donnée".

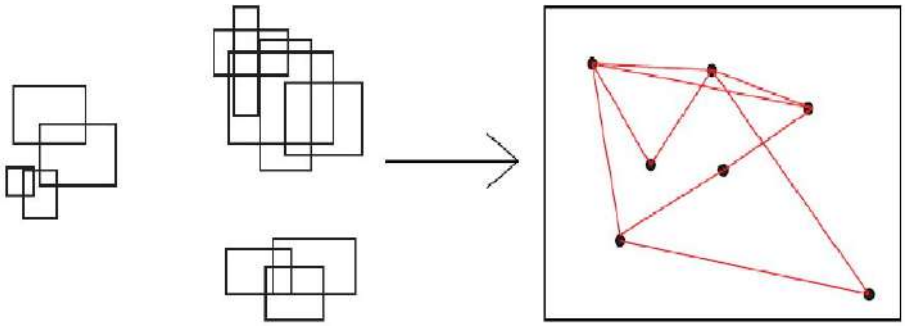


FIGURE 3.18 – Exemple d'initialisation "point".

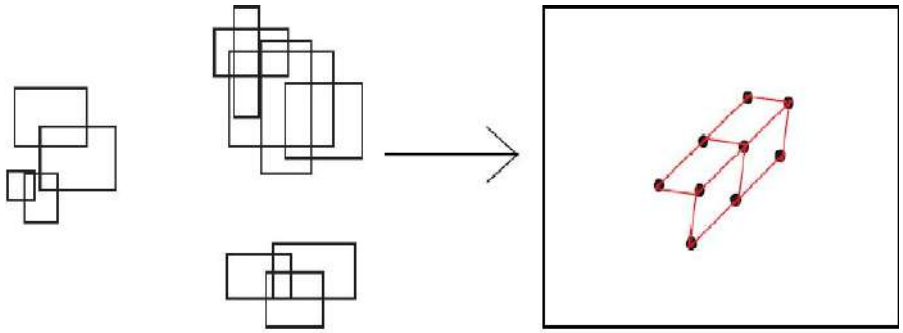


FIGURE 3.19 – Exemple d'initialisation "lininit".

(ACP) à partir de la position des centres des hyper-rectangles des données. La grande différence avec les deux initialisations précédentes est le fait que la carte générée de cette manière est déjà organisée.

Afin de tester ce nouvel algorithme, nous avons créé sept bases de données artificielles présentant diverses difficultés. Nous avons lancé l'algorithme une dizaine de fois sur chaque base de données, puis nous avons calculé l'indice de Jaccard moyen pour vérifier la validité des résultats obtenus.

"2Dim" est constitué de deux groupes, de 200 données chacun, séparés linéairement,

en dimension 2. Chaque groupe se trouve dans un carré de coté 2. Les deux groupes sont de tailles égales mais les intervalles sont de formes et de tailles aléatoires. "3Dim" est composé de quatre classes de même taille disposées sur les sommets d'un tétraèdre dans un espace à 3 dimensions. Dans "5Dim", les classes sont de formes et de tailles différentes dans un espace à 5 dimensions.

Les données "Soleil" sont constitués de cinq classes orientées de différentes manières. Ce jeu contient 195 données en deux dimensions. Le jeu de données "Crochets" est constitué de six groupes d'intervalles en contact. "Croix" est constitué de trois groupes se croisant en un centre, avec un groupe composé de carrés autour de ce centre, un groupe composé d'intervalles allongés verticaux et le dernier d'intervalles allongés horizontaux. Enfin, "Cible" est composé de carrés de même taille, divisés en deux groupes, l'un formant un anneau et le second au centre de cet anneau.

3.3.3.2 Résultats

Les résultats de l'indice de Jaccard montrent que, pour ces données, l'algorithme est capable de retrouver sans erreur le bon nombre de groupes et une segmentation correcte des données. La Figure 3.20 montre qu'en moyenne la mesure de distance d utilisée influence la qualité des résultats obtenus. En particulier la distance L2 semble légèrement moins adaptée que la distance aux sommets et la distance L1, surtout lorsque les groupes sont en contact ("Crochets" et "Croix"). De même, l'initialisation linéaire "lininit" donne en moyenne de meilleurs résultats que les autres initialisations.

La visualisation des groupes obtenus avec une initialisation linéaire et la distance au sommets confirme la qualité de l'algorithme de clustering adapté aux données intervalles(Figures 3.22 à 3.24).

Les résultats obtenus confirment l'efficacité de l'adaptation de S2L-SOM aux données intervalles. Les caractéristiques principales de S2L-SOM sont conservées : le nombre de groupes est déterminé automatiquement, l'algorithme est capable de traiter des groupes de forme non convexe, de plus l'exécution est fiable et rapide. Dans le cas des données intervalles, on remarque que l'algorithme discrimine parfaitement des intervalles de formes différentes, même en cas de contact entre les intervalles et même dans le cas où les centres des intervalles sont confondus. Ces propriétés sont très appréciables pour ce type de données.

Les tests ont montrés que l'algorithme est peu sensible à la distance utilisée pour comparer les données et les prototypes, bien que la distance L2 semble légèrement

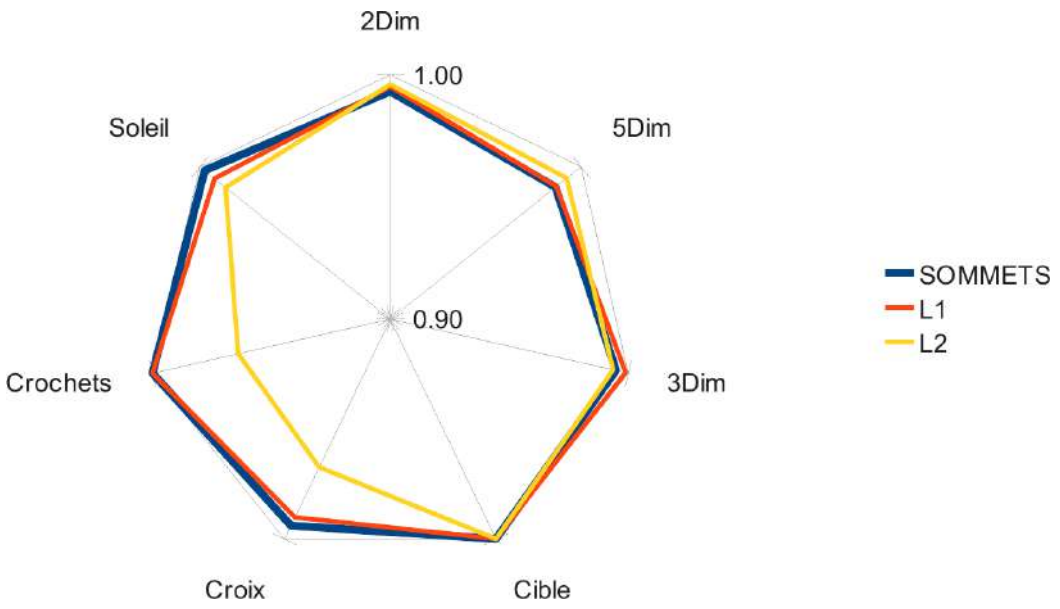


FIGURE 3.20 – Valeurs de l'indice de Jaccard pour différentes bases de données et différentes mesures de distances entre intervalles.

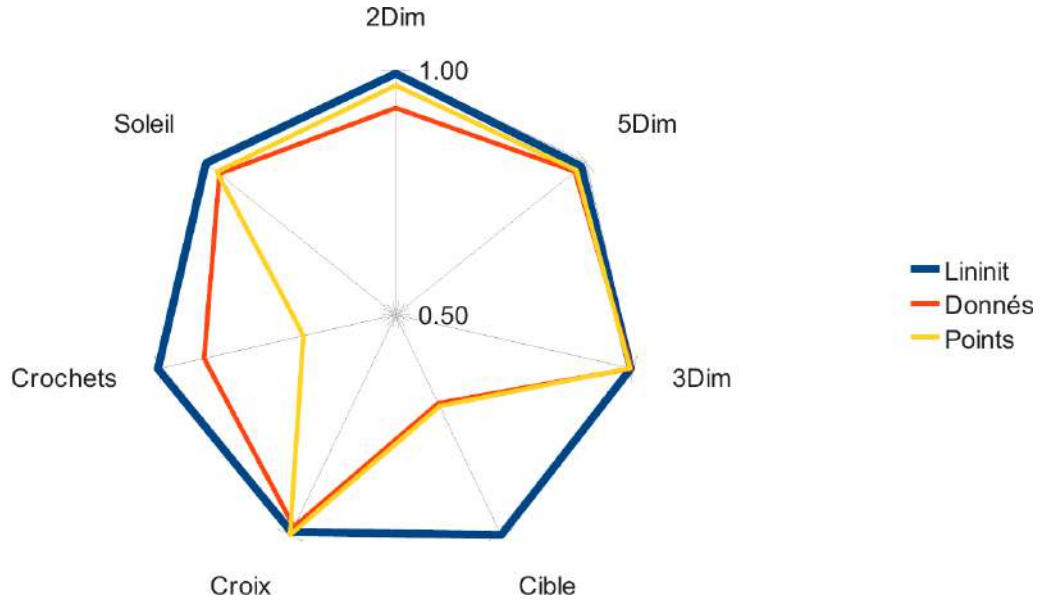


FIGURE 3.21 – Valeurs de l'indice de Jaccard pour différentes bases de données et différentes initialisations de l'algorithme.

moins performante. Par contre, l'initialisation des prototypes joue un rôle important pour la qualité du résultat final, une initialisation linéaire donne de meilleurs résultats (c'est d'ailleurs l'initialisation qui donne les meilleurs résultats en général pour SOM [87]).

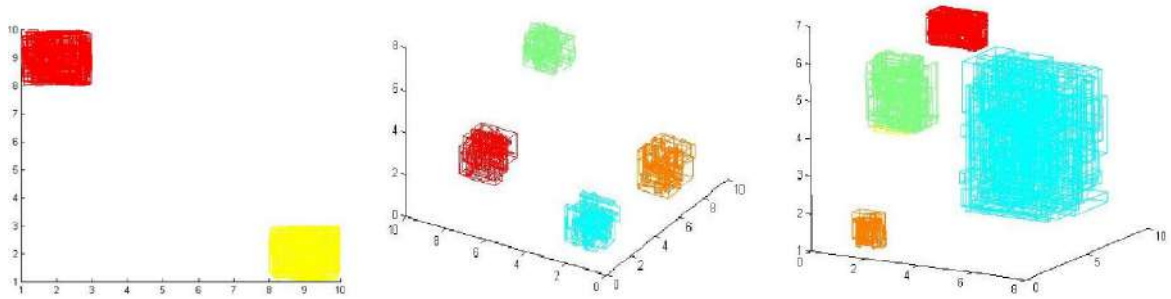


FIGURE 3.22 – Visualisation des groupes obtenus pour les données "2Dim", "3Dim" et "5Dim" (seules les trois premières dimensions ont été représentées).

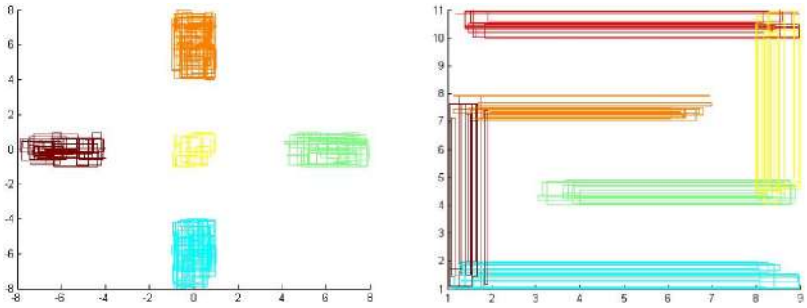


FIGURE 3.23 – Visualisation des groupes obtenus pour les données "Soleil" et "Crochets".

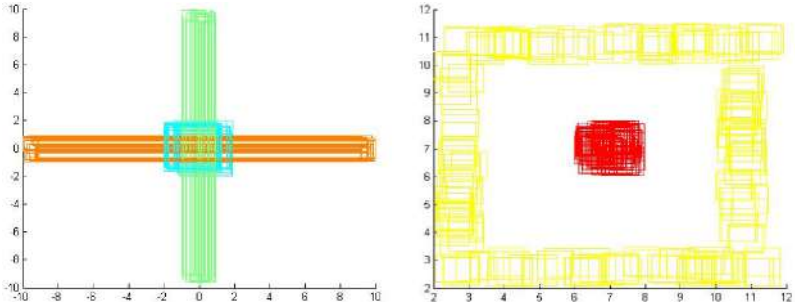


FIGURE 3.24 – Visualisation des groupes obtenus pour les données "Croix" et "Cible".

Une fois une distance et un prototypage définis, l'adaptation de S2L-SOM (et par extension de DS2L-SOM) à d'autres types de données semble donc parfaitement pertinente. De nouvelles adaptations pour l'analyse de données plus complexes sont ainsi fortement envisageables.

3.4 Conclusion

Nous avons montré dans ce chapitre deux extensions possibles de S2L-SOM, l'une permettant d'améliorer la quantification des données par la SOM, l'autre permettant l'analyse de données intervalles. De tels principes peuvent facilement être étendus à d'autre informations estimées pendant l'apprentissage de la SOM (en particulier les informations de densité, qui n'ont pas été utilisées dans ce chapitre), ainsi que pour la résolution d'autres types de problèmes.

Dans le chapitre suivant, nous proposons de nouvelles utilisations d'informations apprises à partir des données pendant l'apprentissage de la carte. L'idée n'est plus d'estimer une partition des données pour représenter leur structure, mais d'estimer une fonction de densité de ces données, qui peut être vu comme un modèle plus précis et plus général de la façon dont les données sont structurées dans leur espace de représentation (c'est à dire de leur distribution).

Chapitre 4

Estimation et comparaison de la distribution des données, application à la sélection de modèles

4.1 Introduction

La croissance exponentielle des données engendre des volumétries de bases de données très importantes. Des études montrent que la quantité des données double tout les trois ans [96]. Toutefois, la capacité à analyser les données reste insuffisante, en particulier parce qu'il existe peu d'algorithmes à la fois performants et rapides.

Par ailleurs, dans de nombreux cas, la capacité à mesurer des similitudes entre différents ensembles de données devient un élément important de l'analyse. Une application importante est l'estimation de la stabilité d'un algorithme de classification ou de tout autre algorithme produisant un modèle des données. Cette stabilité est une mesure des propriétés de généralisation de l'algorithme utilisé, autrement dit de sa capacité à éviter le sur-apprentissage (apprendre les variations aléatoires propres à l'échantillon de données étudié). Une autre application majeure pourrait être l'analyse des flux de données, tel que cela est présenté dans la section 5.4. De nombreuses autres applications sont possibles, telles que la comparaison de grandes bases de données, la fusion de partitions, etc...

Dans ce chapitre, nous proposons donc, dans une première partie, une méthode très rapide d'estimation de la distribution des données à partir d'une SOM enrichie selon les principes exposés dans les chapitres précédents (estimation de la connectivité et de la densité au cours de l'apprentissage de la carte). La distribution est estimée sous la forme d'une fonction de densité, qui peut être utilisée pour l'analyse du phénomène décrit par les données ou pour générer de nouveaux jeux de données de même distribution. Une méthode de visualisation en deux dimensions de cette fonction est aussi présentée dans une seconde partie. La troisième partie présente une mesure de comparaison de la distribution de deux ensembles de données à partir de la similarité entre les deux fonctions de densité associées. La dernière partie du chapitre présente une utilisation de cette comparaison à la sélection de paramètres pour DS2L-SOM selon le principe de stabilité. Le paramètre à sélectionner est la topologie de la carte (nombre de neurones et dimensions de la carte).

4.2 Estimation de la densité des données

La première partie de ce travail porte sur l'estimation de la densité sous-jacente des données par une méthode à deux niveaux. L'idée ici est de pouvoir estimer la distribution des données à partir d'un modèle topologique de ces données. L'apprentissage du

modèle doit être linéaire selon le nombre de données et doit être capable d'apprendre "en ligne", c'est à dire capable d'apprendre à partir d'une seule présentation de chaque donnée, de façon à être utilisable pour l'analyse de grandes bases de données ou de flux de données.

4.2.1 SOM - Enrichie

Nous supposons ici que les données sont décrites sous la forme d'un vecteur numérique d'attributs. Pour commencer, les données sont modélisées à l'aide d'une SOM enrichie, de façon à construire une représentation abstraite de la structure des données. Ensuite, une fonction de densité est estimée à partir de la représentation abstraite.

L'idée est de combiner la réduction de dimension et la vitesse d'apprentissage de la SOM pour construire un nouvel espace de représentation de faible dimension, puis d'appliquer une autre analyse sur cet espace. Ce type de méthode est appelée « méthode à deux niveaux ». Les méthodes à deux niveaux sont connues pour réduire grandement le temps de calcul, les effets du bruit et le « fléau de la dimension » [149, 89].

4.2.1.1 Description de l'algorithme

L'algorithme général procède en trois étapes :

1) La première étape est l'apprentissage de la SOM enrichie. Pendant l'apprentissage, chaque prototype de la SOM est associé à de nouvelles informations extraites des données. Ces informations structurelles seront utilisées durant la deuxième étape pour estimer la fonction de densité. Plus particulièrement, les informations associées aux prototypes sont :
   - *Mode de densité*. C'est une mesure de la densité des données au voisinage du prototype (densité locale). La densité locale est une mesure de la quantité de données présente dans une région restreinte de l'espace de description. Nous utilisons un estimateur à Noyau Gaussien [138] pour cette tâche.
   - *Variabilité locale*. Il s'agit d'une mesure de la variabilité des données représentées par le prototype. Cette variabilité peut être définie comme la distance moyenne entre le prototype et les données qu'il représente.
   - *Le voisinage*. Il s'agit d'une mesure du voisinage du prototype. La valeur de voisinage entre deux prototypes est le nombre de données ayant ces deux prototypes comme meilleurs représentants.

2) La seconde étape est l'estimation de la fonction de densité des données à partir de la SOM enrichie. Cette fonction est estimée à partir des informations associées aux prototypes. Elle est représentée sous la forme d'un modèle de mélange de noyaux Gaussiens sphériques.
3) La dernière étape est la comparaison de deux ensembles de données différents, en utilisant une mesure de dissimilarité capable de comparer les deux fonctions de densité estimées à l'étape précédente.

4.2.1.2 Enrichissement des prototypes

Dans cette étape, certaines informations générales sont extraites à partir des données et stockées dans les prototypes lors de l'apprentissage de la SOM. Dans notre algorithme, les prototypes de la SOM vont être « enrichis » par l'addition de nouvelles valeurs numériques extraites de la structure des données. Cet algorithm est quasiment identique à celui décrit dans la section 2.3.1.1, la différence étant l'ajout de l'estimation de la variabilité locale.

L'algorithme d'enrichissement procède en trois étapes :

Entrée :
- Les données $X = \{x^{(k)}\}_{k=1}^{N}$.

Sortie :
- Une estimation de la densité D_j et de la variabilité locale s_j associées à chaque prototype w_j.
- Une estimation des valeurs de voisinage $v_{i,j}$ associées à chaque paire de prototype w_i et w_j.

1) **Initialisation** :
   - Initialisation des paramètres de la SOM
   - $\forall i, j$ les densités locales (D_j), les valeurs de voisinages ($v_{i,j}$), les variabilités locales (s_j) et le nombre de données représentées par w_j (N_j) sont initialisés à zero.
2) **Tirage aléatoire d'une donnée** $x^{(k)} \in X$:
   - Calcul de $d(w_j, x^{(k)})$, la distance euclidienne entre la donnée $x^{(k)}$ et chaque prototype w_j.
   - Recherche des deux meilleurs prototypes (BMUs : Best Match Units) w_{u^*} et $w_{u^{**}}$:

$$u^* = \arg\min_i(d(w_i, x^{(k)})) \quad \text{et} \quad u^{**} = \arg\min_{i \neq u^*}(d(w_i, x^{(k)}))$$

.

3) **Mise à jour des informations structurelles** :
   - Variabilité :

$$s_{u^*}(t) = s_{u^*}(t-1) - \varepsilon(t)r(t)\left(s_{u^*}(t-1) - d(w_{u^*}, x_k)\right)$$

   - Densité :

$$\forall j, D_j(t) = D_j(t-1) - \varepsilon(t)r(t)\left(D_j(t-1) - e^{-\frac{\|x^{(k)} - w_j(t)\|^2}{2\sigma^2}}\right)$$

   - Voisinage :

$$\nu_{u^*u^{**}}(t) = \nu_{u^*u^{**}}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*u^{**}}(t-1) - 1\right)$$
$$\nu_{u^*i}(t) = \nu_{u^*i}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*i}(t-1)\right) \quad \forall i \text{ voisin de } u^*$$

   Avec $\varepsilon(t)$ le taux d'apprentissage et $r(t) = \frac{1}{1+e^{\left(-\frac{t}{t_{max}}\right)}}$.

4) **Mise à jour des prototypes de la SOM** w_i comme défini dans [87].

5) **Répéter** T **fois les étapes** 2 **à** 4**, jusqu'à ce que** $t = t_{max}$.

Dans cette étude nous avons utilisé les paramètres par défaut de la SOMToolbox [151] pour l'apprentissage de la SOM et nous utilisons $T = max(N, 50 \times M)$ comme dans [151]. Le nombre M de prototypes ne doit être ni trop faible (la SOM ne modélise pas bien les données), ni trop grand (coûteux en temps de calcul). Choisir M proche de $\sqrt{N}$ semble être un bon compromis [151]. Enfin, σ est défini comme la distance moyenne entre un prototype et son plus proche voisin.

À la fin de cette étape, à chaque prototype est associée une valeur de densité et de variabilité, et à chaque paire de prototypes est associée une valeur de voisinage. De ce fait, une grande partie de l'information sur la structure des données est stockée dans ces valeurs. Il n'est plus nécessaire de garder les données en mémoire.

4.2.1.3 Version Batch

La version Batch de l'algorithme est adaptée aux données complexe pouvant être décrites par l'intermédiaire d'une matrice de dissimilarité. Cependant, cet algorithme ne fonctionne pas en ligne, et il est nécessaire de conserver en mémoire les similarités entre les données et les prototypes. Il n'est donc pas adapté à l'analyse de grandes bases de données ou de flux.

Algorithme :

1) **Apprentissage des prototypes** :
   - Les prototypes sont appris selon un algorithme au choix (par exemple la version Batch de SOM, Neural Gaz ou une méthode à noyaux).
   - L'algorithme retourne $d(W, X)$, la matrice des dissimilarités entre chaque prototype et chaque donnée.
2) **Mise à jour des informations structurelles** :
   Soit $X_j = \{x_k \in X | j = \arg\min_n d(w_n, x_k)\}$ l'ensemble des données affectés au prototypes w_j. On calcule les informations suivantes :
   - $N_j = |X_j|$ le cardinal de X_j.
   - Variabilité : $s_j = \frac{1}{N_j} \sum_{x_k \in X_j} d(w_j, x_k)$.
   - Densité : $D_j = \frac{1}{N} \sum_{x_k \in X} e^{-\frac{d(w_j, x_k)^2}{2\sigma^2}}$.
   - Voisinage : $v_{i,j} = \frac{1}{N}|X_{ij}|$ avec :

$$X_{ij} = \{x_k \in X_i \cup X_j | i = \arg\min_{n \neq j} d(w_n, x_k) \text{ et } j = \arg\min_{n \neq i} d(w_n, x_k)\}$$

Un avantage important de la version Batch est que les informations structurelles sont plus fiables que dans la version stochastique, puisqu'elles sont calculées seulement une fois que les prototypes aient été estimés de façon optimale.

4.2.2 Estimation de la fonction de densité

L'objectif de cette étape est d'estimer la fonction de densité qui associe une valeur de densité à chaque point de l'espace de représentation des données. Nous connaissons la valeur de cette fonction au niveau des prototypes (D_i). Il faut en déduire une approximation de la fonction.

Nous faisons ici l'hypothèse que cette fonction peut être correctement approximée sous la forme d'un mélange de noyaux Gaussiens sphériques ($\{K_i\}_{i=1}^M$), où K_i est une fonction Gaussienne centrée sur un prototype w_i et M est le nombre de prototype. La fonction de densité peut alors s'écrire :

$$f(x) = \sum_{i=1}^{M} \alpha_i K_i(x)$$

avec

$$K_i(x) = \frac{1}{\sqrt{2\pi}.h_i} e^{-\frac{|w_i - x|^2}{2{h_i}^2}} \text{ et } \sum \alpha_i = 1$$

La méthode la plus populaire pour estimer un modèle de mélange (C'est à dire trouver h_i et α_i) est l'algorithme EM (Expectation-Maximization, [37]). Cependant, cet algorithme travaille dans l'espace des données. Ici nous avons seulement à disposition la SOM enrichie au lieu de l'ensemble des données et nous ne pouvons pas utiliser l'algorithme EM (nous avons montré au chapitre 2 que les prototypes seuls ne sont pas toujours pertinents pour la représentation de la distribution des données).

Nous proposons donc une heuristique pour choisir h_i :

$$h_i = \frac{\sum_j \frac{v_{i,j}}{N_i+N_j}(s_i N_i + d_{i,j} N_j)}{\sum_j v_{i,j}}$$

où N_i est le nombre de données représentées par le prototype w_i, s_i est la variabilité de w_i et $d_{i,j}$ est la distance euclidienne entre w_i et w_j. L'idée est que h_i est l'écart type des données représentées par K_i. Ces données sont aussi représentées par w_i et ses voisins. Ainsi h_i dépend de la variabilité s_i calculée pour w_i et les distances $d_{i,j}$ entre w_i et ses voisins, pondérées par le nombre de données représentées par chaque prototypes et par la connectivité entre w_i et ses voisins.

Puisque la densité D au niveau des prototypes w est connue ($f(w_i) = D_i$), on peut déterminer les pondérations α_i. Ces pondérations sont solution du système d'équations linéaires suivant :

$$D = \sum_{i=1}^{M} \alpha_i K_i(w)$$

avec

$$D = [D_j]_{j=1}^{M} \text{ et } w = [w_j]_{j=1}^{M}$$

Cependant, il existe une infinité de solutions à cette équation, ce qui rend impossible toute résolution matricielle basée sur une inversion de matrice. De plus, la solution obtenue par calcul de la pseudo-inverse [13] n'est souvent pas satisfaisante, en particulier parce qu'elle peut contenir des valeurs de α négatives qui ne garantissent plus la contrainte : $\forall x, f(x) > 0$. Nous utilisons donc pour résoudre ce système une méthode très simple de descente de gradient. Les α_i sont initialisés par les valeurs de D_i, puis voient leur valeur réduite progressivement (avec une valeur minimum de 0) jusqu'à satisfaire au mieux $D = \sum_{i=1}^{M} \alpha_i K_i(w)$. Ainsi les valeurs de α restent en moyenne proportionnelles aux valeurs de D_i, ce qui satisfait l'hypothèse que chaque densité D_i est générée principalement par le prototype w_i. Pour cela, nous optimisons le critère suivant :

$$\alpha = \arg\min_{\alpha} \frac{1}{M} \sum_{i=1}^{M} \left[\sum_{j=1}^{M} (\alpha_j K_j(w_i)) - D_i \right]^2$$

Algorithme :

1) **Initialisation** : $\forall i, \alpha_i = D_i$
2) **Calcul de l'écart** : $\forall i, ecart(i) = \sum_{j=1}^{M} \alpha_j K_j(w_i) - D_i$
3) **Mise à jours des coefficients** :
   $\forall i, \alpha_i(t) = max\,[0\,;\, \alpha_i(t\text{-}1) - \epsilon * ecart(i)]$, avec ϵ le pas du gradient. Nous utilisons ici $\epsilon = 0.1$.
4) **Tant que** $moyenne(|ecart|) > seuil$: retourner en 2, sinon retourner les α_i. Le seuil est choisi par l'utilisateur, nous choisissons ici 1% de la densité moyenne.

Ainsi, nous obtenons une fonction de densité qui est un modèle des données représentées par la SOM.

4.2.3 Validation

4.2.3.1 Complexité de l'algorithme

La complexité de l'algorithme est $O(T \times M)$, avec T le nombre d'étape de l'apprentissage de la SOM enrichie et M le nombre de prototypes. Il est recommandé de choisir $T > 10 \times M$ pour une bonne convergence de la SOM [151]. Pour cette étude nous choisissons $T = max(N, 50 \times M)$ comme dans [151], avec N le nombre de données. Cela implique que si $N > 50 \times M$ (grande base de données), la complexité de l'algorithme est $O(N \times M)$, c'est à dire qu'elle est linéaire en N pour une taille fixée de la SOM. Ainsi le processus est très rapide et donc adapté au traitement de grandes bases de données. De plus les très grandes bases de données peuvent être traitées en fixant $T < N$ (ce qui revient à travailler sur un échantillon aléatoire de la base).

Notre algorithme est donc plus rapide que les méthodes traditionnelles comme l'estimateur à noyaux [138], qui nécessite de plus de conserver toute la base en mémoire, ou les methodes basées sur l'algorithme EM (par exemple la Generative Topographic Mapping [16], le Mélange de Noyaux Gaussien [48] ou la Generative-SOM [150]) puisque la vitesse de convergence peut être extrêmement lente [124, 117].

4.2.3.2 Validation visuelle

Les figures 4.1 à 4.4 montrent des exemples de densités estimées de cette façon.

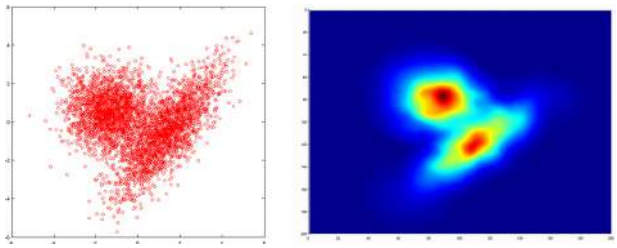


FIGURE 4.1 – Données « Engytime » (gauche) et la fonction de densité estimée (droite).

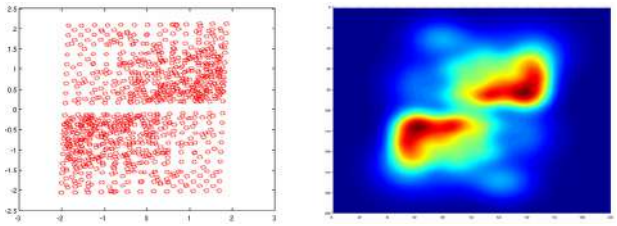


FIGURE 4.2 – Données « Wingnut » (gauche) et la fonction de densité estimée (droite).

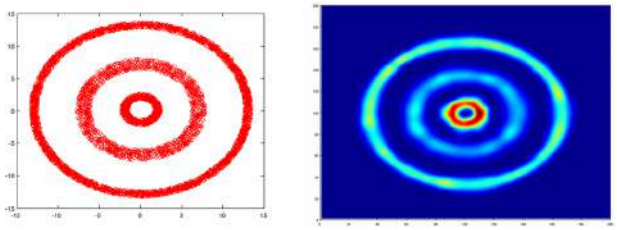


FIGURE 4.3 – Données « Rings » (gauche) et la fonction de densité estimée (droite).

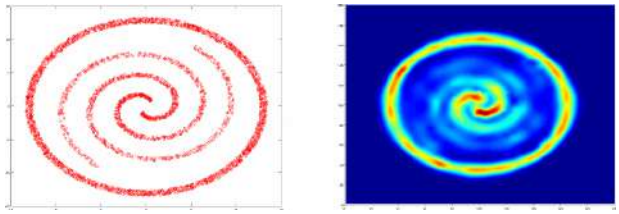


FIGURE 4.4 – Données « Spirals » (gauche) et la fonction de densité estimée (droite).

4.3 Un outil de visualisation de la structure intra et inter groupes

Dans cette section, nous introduisons un outil de visualisation de la SOM enrichie segmentée par DS2L-SOM (ou par un autre algorithme) et de la fonction de densité estimée à partir de la SOM. Cette visualisation permet de mettre en valeur la structure intra et inter groupe des données.

4.3.1 Description de l'outil de visualisation utilisé

La SOM enrichie segmentée est accompagnée d'un ensemble d'informations qui peut être utilisé pour compléter l'analyse des données, telles que la matrice des distances entre prototypes et la matrice de densité, mais aussi les valeurs des connexions qui peuvent être utilisées pour déterminer l'importance relative de chaque prototype pour la représentation des données. Il est possible de représenter toutes ces informations en une seule image permettant une analyse fine de la structure de chaque groupe et de leurs relations :

- Les prototypes sont projetés dans un espace à deux dimensions (éventuellement trois) à l'aide d'une projection de Sammon, qui conserve au mieux les distances initiales entre prototypes [129].
- La taille des disques représentant les prototypes est proportionnelle à la densité associée à chaque prototype.
- La couleur de chaque prototype dépend du cluster auquel il est associé.
- Les connexions de voisinage (topologie locale) sont représentées par un segment reliant les prototypes voisins.
- Les variations de la fonction de densité sont représentés sous la forme de courbes de niveaux.

Cette visualisation permet d'obtenir des informations à la fois sur la structure inter groupes (nombre de groupes, similarités entre les groupes) mais aussi la structure intra groupe (topologie locale, densité locale et variation de densité au sein du groupe, variabilité des données représentées).

4.3.2 Applications

Nous avons appliqué ce procédé à huit bases de données artificielles et réelles. Les figures 4.5 à 4.9 montrent les visualisations pouvant être obtenues à partir de données

artificielles de faible dimension.

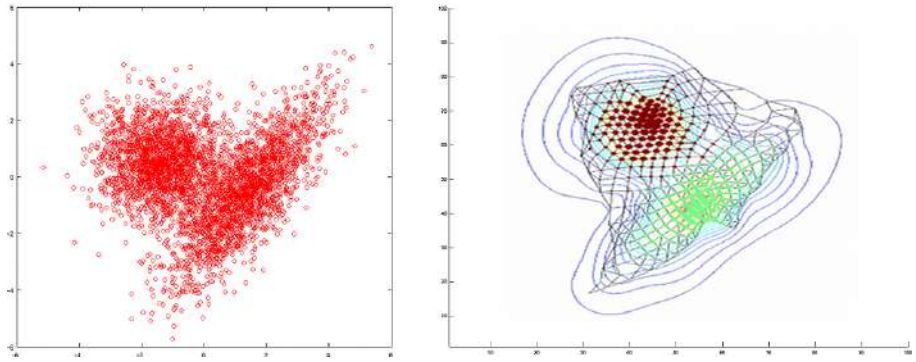


FIGURE 4.5 – Visualisation des données « Engytime ».

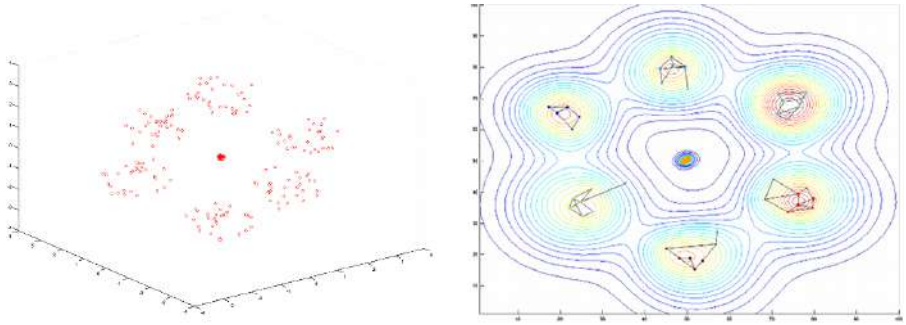


FIGURE 4.6 – Visualisation des données « Hepta ».

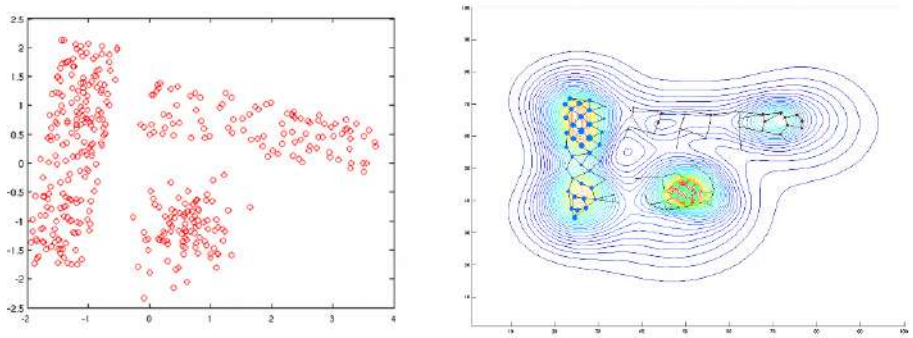


FIGURE 4.7 – Visualisation des données « Lsun ».

On remarque que la structure des données est bien conservée par l'algorithme de quantification et de classification et qu'elle est bien représentée par le procédé de visualisation. La densité des données est facilement identifiable par la taille de la représentation des prototypes et par les lignes de niveaux. Ces dernières permettent de plus

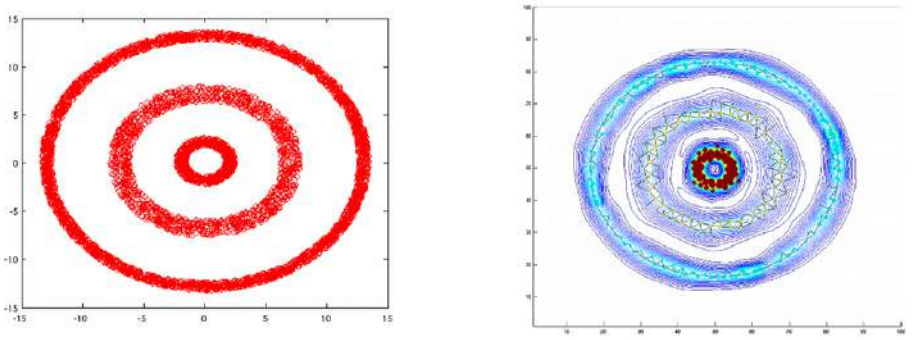


FIGURE 4.8 – Visualisation des données « Rings ».

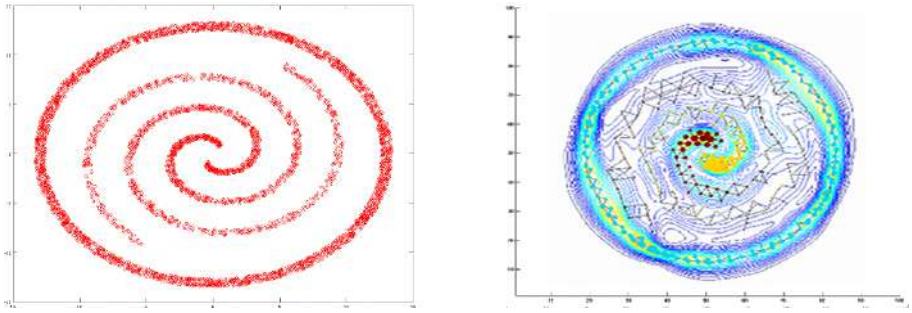


FIGURE 4.9 – Visualisation des données « Spirales ».

de visualiser en deux dimensions la forme générale des différents clusters et leur taille relatives. La visualisation des connexions, ainsi que les différentes couleurs associées aux prototypes, autorisent une description visuelle de la segmentation des données en différents clusters. Deux clusters fortement connectés sont représentatifs de groupes de données en contact, comme dans la figure 4.1, alors que l'absence de connexion dénote des groupes bien séparés dans l'espace de représentation des données (par exemple dans la figure 4.2). De plus, la visualisation est suffisamment fine pour permettre une représentation des données de distribution complexe, comme cela est illustré par les figures 4.3 et 4.4.

Les figures 4.10 à 4.12 montrent quelques exemples de visualisations pouvant être obtenues à partir de données réelles. Les données « Iris » sont une description selon quatre modalités de fleurs de trois espèces différentes. Les données « Fourmis » décrivent l'activité de chaque individus d'une colonie de fourmis (11 variables). Enfin les données « Enfants » sont une description du temps passé dans différentes activités de jeu dans un groupe d'enfants (8 variables).

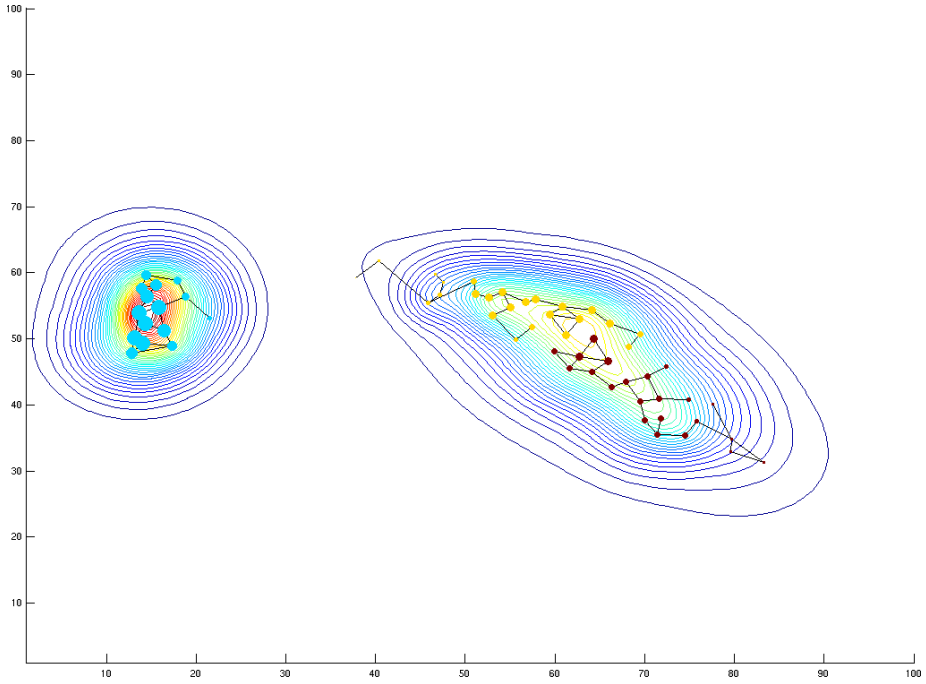


FIGURE 4.10 – Visualisation des données « Iris ».

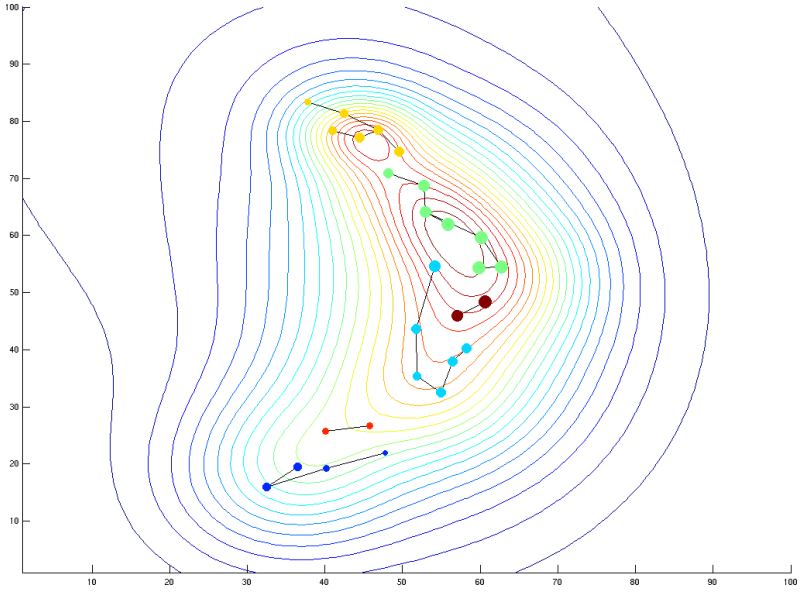


FIGURE 4.11 – Visualisation des données « Fourmis ».

La visualisation de ces bases de données de petite taille mais de dimension supérieure à trois illustre la capacité du procédé de visualisation à projeter les informations pertinentes dans un espace à deux dimensions. Ainsi, les données “Iris” (Figure 4.10)

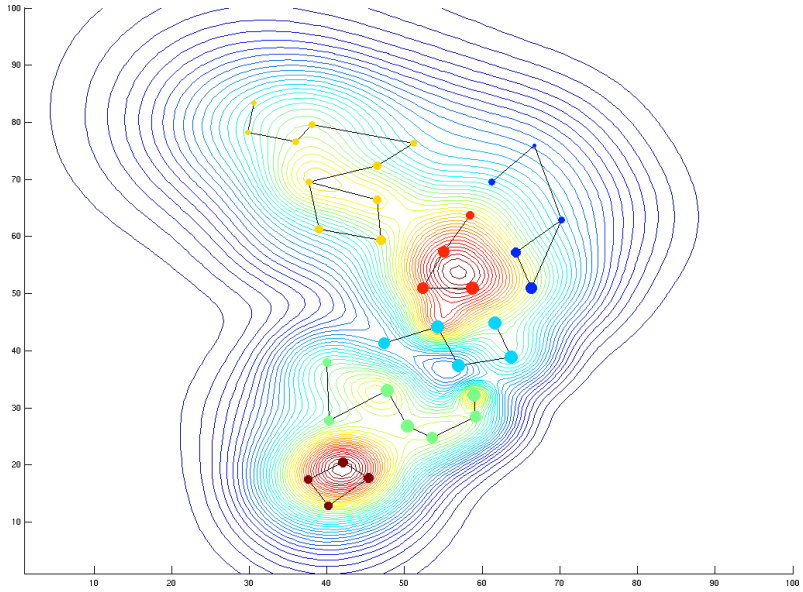


FIGURE 4.12 – Visualisation des données « Enfants ».

sont structurées en deux groupes bien distincts, un de ces deux groupes étant lui-même subdivisé en deux groupes très proches. Les trois groupes sont découverts automatiquement par l'algorithme de classification et correspondent à trois espèces distinctes de fleurs. Pour les données "Fourmis" (Figure 4.11), chaque cluster détecté par l'algorithme correspond à un comportement et un rôle social différent au sein de la colonie (Chasseuses, nourrices, nettoyeuses, gardes, etc...). Ici, on n'observe pas de séparations nette en terme de densité entre les groupes, ce qui signifie que certains comportements intermédiaires sont possibles. L'existence de ces comportements intermédiaires sont connus en biologie, en particulier grâce à la présence de fourmis généralistes capables d'effectuer n'importe quelle tâche en fonction des besoins de la colonie [voir 72]. Enfin, les données "Enfants" (Figure 4.12) représentent les activités de jeux d'enfants de maternelles en récréation. Les données sont divisées en deux ensembles de densité assez bien séparés, chacun subdivisé en deux sous-ensembles. Le sous-groupe central a lui-même été subdivisés en trois clusters par l'algorithme. Il est intéressant de noter que, globalement, l'ordre des groupes de haut en bas correspond à une augmentation de l'âge des enfant et une augmentation de la complexités des activités de jeux. Le groupe jaune est pratiquement uniquement composé d'enfant en première année de maternelle, alors que la grande majorité des enfant en dernière année sont dans le groupe marron. La subdivision des deux années intermédiaires en quatre clusters dénote des différences individuelles dans la dynamique du développement des enfants. La baisse de densité entre le groupe bleu clair et le groupe vert sépare les enfant passant le plus clair de leur temps en jeux sociaux (avec leurs congénères) des enfants jouant le plus souvent seul.

Cela indique qu'un enfant qui commence à jouer avec ses congénères ne reviendra plus, ou rarement, à des jeux solitaires. L'ensemble de ces informations semblent en accords avec les connaissances du domaine [cf. 56].

4.4 Une mesure pour la comparaison de distributions

Nous proposons ici une mesure heuristique de dissimilarité entre modèles de clusters, basée sur la comparaison des ensembles de prototypes qui décrivent ces clusters et leur structure. Nous supposons ici que ces prototypes ont été préalablement calculés à partir des données par l'algorithme présenté dans la section 4.2. L'objectif de cette mesure est de pouvoir comparer les distributions de deux ensembles de données décrites dans les même espace, de façon à détecter si leur distributions sont identiques, similaires ou nettement différentes (cf. Figure 4.13). Une application de ce type de comparaisons pourrait être la détection de changement dans la structure d'un flux de données.

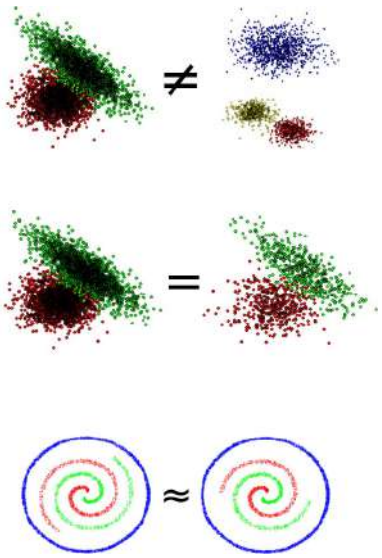


FIGURE 4.13 – Exemple de comparaisons de la distribution des données.

4.4.1 Mesure de dissimilarité

Il est possible de définir une mesure de dissimilarité entre deux ensembles de données A et B, représenté chacun par une SOM et une fonction de densité :

$$SOM_A = \left[\{w_i^A\}_{i=1}^{M^A}, f^A\right]$$

et

$$SOM_B = \left[\{w_i^B\}_{i=1}^{M^B}, f^B\right]$$

Avec M^A et M^B le nombre de prototypes des modèles SOM_A et SOM_B, et f^A et f^B les fonctions de densité de A et B calculées à la section 4.2.2.

La dissimilarité entre A et B est :

$$\begin{aligned} CBd(A,B) &= \frac{\sum_{i=1}^{M^A} f^A(w_i^A) log\left(\frac{f^A(w_i^A)}{f^B(w_i^A)}\right)}{M^A} + \frac{\sum_{j=1}^{M^B} f^B(w_j^B) log\left(\frac{f^B(w_j^B)}{f^A(w_j^B)}\right)}{M^B} \\ &= CBd_A + CBd_B \end{aligned}$$

L'idée est de comparer les fonctions de densité f^A et f^B pour chaque prototype w de A et B. Si les distributions sont identiques, ces deux valeurs doivent être très proches.

Cette mesure est une adaptation de l'approximation pondérée de Monté Carlo de la mesure symétrique de Kullback–Leibler (voir [69]), en utilisant les prototypes de la SOM comme un échantillons de l'ensemble des données. L'idée est de comparer pour chaque prototype i d'un modèle la densité D_i estimée à partir des données et la densité théorique au niveau de ce prototype, $Fd(w_i)$, estimée par la fonction de densité de l'autre modèle. Si les modèles sont identiques, ces deux mesures de densité doivent être très proches.

En outre, cet indice vérifie bien les propriétés d'une mesure de dissimilarité :

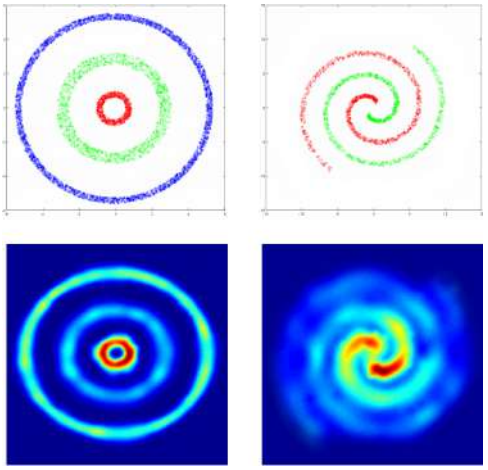
- Positivité : $CBd(A,B) > 0$
- Symétrie : $CBd(A,B) = CBd(B,A)$
- Séparation : $CBd(A,A) = 0$ par construction.

4.4.2 Validation

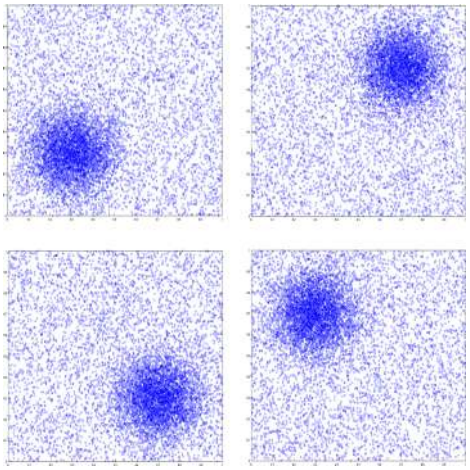
4.4.2.1 Description des distributions de données utilisées

De façon à démontrer les performances de la mesure de dissimilarité proposée, nous avons utilisé neuf générateurs de données artificielles et une base de données réelles.

Les générateurs « Ring 1 », « Ring 2 », « Ring 3 », « Spiral 1 » et « Spiral 2 » génèrent cinq types de jeux de données non-convexes en deux dimensions, de densité et de variance différentes. « Ring 1 » est un anneau de rayon 1 (forte densité), « Ring 2 » un anneau de rayon 3 (faible densité) et « Ring 3 » un anneau de rayon 5 (densité moyenne). « Spiral 1 » et « Spiral 2 » sont deux spirales parallèles. La densité des points dans les spirales diminue avec le rayon. « Noise 1 » à « Noise 4 » sont des distributions en deux dimensions, chacune composée d'une distribution Gaussienne accompagnée d'un bruit homogène très important. Des données des différentes distributions peuvent être générées aléatoirement à volonté (Figure 4.4.2.1).



(a) Visualisations des données « Rings » et « Spirals » (en haut) et de leur fonction de densité estimée par notre algorithme (en bas).



(b) Visualisations des données « Noises ».

FIGURE 4.14 – Exemples de jeux de données pouvant être générés pour l'expérience.

Pour finir, la base de donnée « Shuttle » vient du UCI repository. Il s'agit d'une base de données à neuf dimensions avec 58000 instances. Les données sont divisées en sept classes. À peu prés 80% des données appartiennent à la classe 1.

4.4.2.2 Validité de la mesure de dissimilarité

Il est impossible de démontrer que deux distributions sont exactement identiques. Cependant, une faible dissimilarité entre deux modèles n'est possible que si les deux distributions sont similaires, et cela nous donne une bonne indication sur la similarité entre les deux distributions des données. D'autre part, une grande dissimilarité indique que les données proviennent de distributions différentes. Ainsi, si notre mesure de dissimilarité est performante, il devrait être possible de comparer différentes bases de données (ayant les mêmes attributs) et de détecter la présence de distributions similaires. La dissimilarité des données générées selon la même loi de distribution doit être bien plus faible que la dissimilarité entre données générées selon des distributions très différentes.

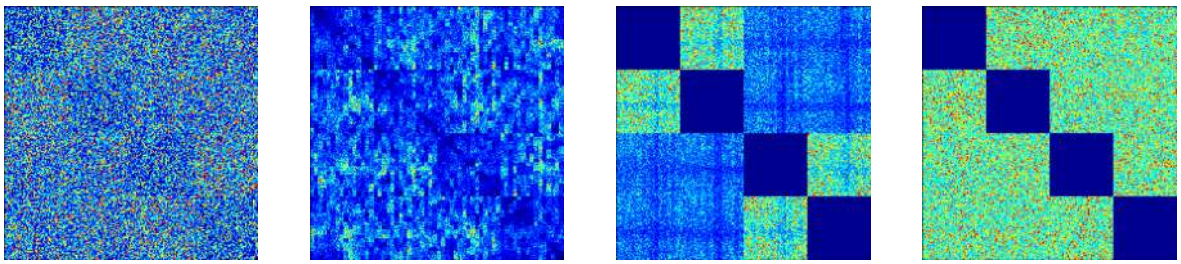
Pour vérifier cette hypothèse nous avons appliqué le protocole suivant :

1) Nous avons généré 250 jeux de données de distribution « Ring 1 », « Ring 2 », « Ring 3 », « Spiral 1 » et « Spiral 2 » (50 de chaque). Ces jeux contiennent entre 500 et 50000 données.
2) Pour chacun de ces jeux de données, nous avons appris une SOM enrichie, de façon à obtenir un ensemble de prototypes représentatif de ces données. Le nombre de prototypes varie entre 50 et 500.
3) Une fonction de densité à été estimée pour chaque SOM et toutes ces fonctions ont été comparées les unes aux autres selon la mesure de dissimilarité proposée.
4) Pour finir, chaque SOM a été étiquetée en fonction de sa distribution (les étiquettes sont « ring 1 », « ring 2 » , « ring 3 » , « spiral 1 » et « spiral 2 »). Puis nous avons calculé un indice de compacité et de séparabilité des groupes de SOM de même étiquette à l'aide de l'indice généralisé de Dunn [40]. Cet indice est d'autant plus grand que les objets ayant la même étiquette sont similaires entre eux et sont dissimilaires aux objets d'étiquettes différentes, en fonction de la mesure de dissimilarité utilisée.

Nous avons utilisé le même protocole avec les distributions « Noise 1 » à « Noise 4 », et avec la base de données « Shuttle ». Deux types de distributions ont été extraites de la base « Shuttle », en utilisant un sous-échantillonnage aléatoire (tirage avec remise) des données de la classe 1 (« Shuttle 1 ») et des données des autres classes (« Shuttle 2 »).

Nous avons comparé nos résultats avec certaines mesures de dissimilarité, basées sur la distance (ici Euclidienne), généralement utilisées pour comparer deux ensembles de

données (ici on compare les deux séries de prototypes des deux SOM). Ces mesures sont la distance moyenne (Ad : la distance moyenne entre toutes les paires de prototypes dans les deux SOM), la distance minimale (Md : la plus petite distance entre les deux ensembles de prototypes) et la distance de Ward (Wd : la distance entre les deux centroïdes, avec une pondération en fonction du nombre de prototypes dans les deux SOM, [77]). Tous les résultats sont présentés dans la Table 4.1. Les valeurs de l'indice de Dunn obtenues à partir des différentes mesures montrent que la mesure de dissimilarité proposée (CBd) est plus efficace que les mesures basées sur la distance (en accord avec l'inspection visuelle de la matrice de dissimilarité, voir Figure 4.15 pour un exemple). Ce résultat souligne l'efficacité de notre mesure de dissimilarité pour la détection de sous-ensembles similaires dans des bases de données différentes, et ce même si les sous-ensembles en question sont de tailles nettement différentes. En effet, la valeur de l'indice de Dunn pour notre mesure est bien plus grande que pour les mesures basées sur la distance. Cette mesure, basée sur la densité, est donc plus efficace que les mesures utilisant les distances entre prototypes. Ce résultat est valable pour les trois types de distributions testées : données non-convexes, données bruitées et données réelles.



(a) Ad (b) Md (c) Wd (d) CBd

FIGURE 4.15 – Visualisations de la matrice de dissimilarité obtenue entre différents jeux de données de distribution « Noise » 1 à 4. Les cellules sont d'autant plus sombre que la similarité est élevée. Les jeux de données sont triés selon leur distribution, ainsi les comparaisons de jeux de même distribution apparaissent selon quatre carrés sur la diagonale. Cette diagonale doit être la plus sombre possible pour une bonne performance de la mesure.

On peut noter cependant que les performances sont moins bonnes pour les données convexes que pour les autres. Cela s'explique par le fait que ces données sont peu adaptées à une modélisation par mélange de Gaussiennes. Une bonne approximation de la fonction nécessite un nombre suffisant de neurones. Or ici le nombre de prototypes est fixé (entre 50 et 500) et les modèles de moins de 150 prototypes environ ne peuvent pas donner une bonne approximation des distributions "Rings" et "Spirals". On voit bien que le choix du nombre de neurones est critique pour l'obtention de bon résultats.

TABLE 4.1 – Valeurs de l'indice de Dunn obtenues à partir de diverses mesures de dissimilarité pour comparer différentes distributions de données.

Distributions à comparer	Ad	Md	Wd	**CBd**
Ring 1 à 3 + Spiral 1 et 2	0.4	0.9	0.5	**1.6**
Noise 1 à 4	1.1	1.4	22.0	**115.3**
Shuffle 1 et 2	1.1	16.5	6.3	**27.6**

Le choix de ce paramètre par l'utilisation de méthodes basées sur la stabilité est présenté dans la section 4.5.

Lorsque le nombre de prototypes est suffisant, les différentes distributions sont parfaitement différenciées. Comme on peut le voir sur la figure 4.16, les distances entre modèles correspondant à une même distribution sont nettement plus faibles que les distances entre modèles de distributions différentes. Pour visualiser les similarités entre modèles, nous avons utilisé une projection de Sammon [129]) en deux dimensions, qui respecte au mieux les similarités entre éléments dans l'espace de projection.

Une application de cette mesure de dissimilarité pourrait être la détection de variation dans la structure d'un flux de données.

A titre d'exemple, nous avons présenté au système des jeux aléatoires de 500 données de type « Spirale 1 » jusqu'au temps 5, puis nous avons présenté des données de type « Anneau 5 » jusqu'au temps 20, puis de type « Spirale 2 » jusqu'au temps 25 et pour finir de type « Spirale 1 » à nouveau jusqu'au temps 30. Le système apprend pour chaque jeu de données reçu une SOM enrichie et la compare à celle du jeu précédent.

La figure 4.17 représente la différence entre les deux modèles de deux jeux consécutifs au cours du temps. Les variations dans la structure du flux sont parfaitement détectées par le système. En effet, lorsque le flux ne varie pas, les modèles correspondants sont très proches et la mesure de dissimilarité donne une valeur très faible. Au contraire, si la structure du flux varie, les modèles correspondants sont nettement moins similaires, et la mesure de dissimilarité est nettement plus élevée.

4.4.3 Extention aux comparaisons de clusters

L'extension de la méthode à la comparaison de segmentation est naturelle. Nous disposons en effet d'un outil adapté de segmentation de l'ensemble des prototypes :

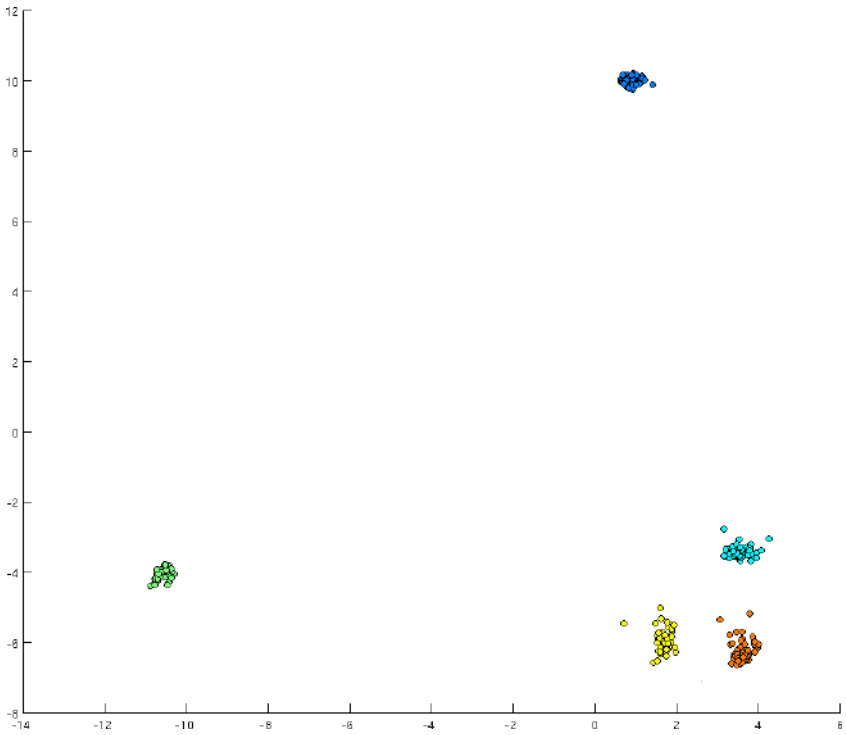


FIGURE 4.16 – Visualisations des similarités entre modèles (un point = un modèle). Bleu : modèles de l'Anneau 5, Vert : Anneau 1, Turquoise : Anneau 3, Jaune et Orange : Spirales 1 et 2.

DS2L-SOM. L'idée est donc de détecter, après clustering des deux bases de données, si certains clusters sont communs, si certains sont similaires avec quelques variations (dérive de concept) et si certains clusters sont propres à chaque base :

1) Définir la fonction de densité de chaque cluster C_k des bases A et B :

$$f_k(x) = \sum_{i \in C_k} \alpha_i K_i(x))$$

2) Pour chaque cluster k appartenant à une des bases, trouver le cluster k' le plus similaire dans l'autre base :

$$k' = \arg\min_i CBd(k, i)$$

3) Pondérer la distance entre k et k' par la distance moyenne entre k et les autres clusters de la même base de données que k.

$$CBd_{pond}(k, k') = \frac{CBd(k, k')}{\frac{1}{|C_k|} \sum_{i \in C_k, i \neq k} CBd(k, i)}$$

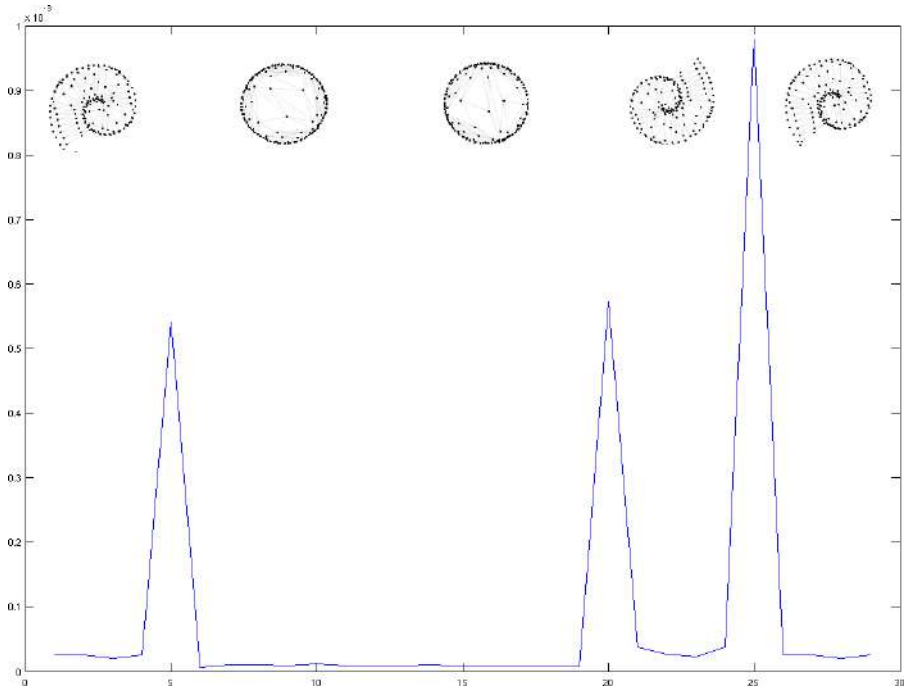


FIGURE 4.17 – Dissimilarité entre les deux modèles de deux jeux consécutifs au cours du temps. Certains modèles ont été représentés pour illustrer les variations temporelles. Ces variations de la structure du flux (temps 5, 20 et 25) sont parfaitement détectées.

Remarque : CBd_{pond} n'est pas symétrique.

4) Il est alors possible de comparer les bases de données :
   - Si $CBd_{pond}(k, k') < 1$, il s'agit d'une dérive de concept entre k et k'.
   - Si $CBd_{pond}(k, k') \approx 0$, k et k' sont les même clusters.
   - Si $CBd_{pond}(k, k') \geq 1$, k et k' sont des clusters différents.

De cette façon il est facile de détecter des clusters communs aux deux bases et des clusters propres à chacune.

Il est aussi possible de détecter des fusions ou des fissions de clusters d'une base à l'autre :

1) Décomposer l'indice : $CBd(k, k') = CBd_k + CBd_{k'}$ comme en 4.4.1.
2) Définir un indice de fusion-fission entre k et k' :

$$F(k, k') = \frac{CBd_k}{CBd_{k'}}$$

3) Analyser F :
   - Si $F(k, k') \approx 1$, les deux clusters sont bien séparés ou très similaires.
   - Si $F(k, k') \ll 1$, k est une partie du cluster k'.
   - Si $F(k, k') \gg 1$, k' est une partie du cluster k.

Ainsi si deux clusters k_1 et k_2 d'une base A ont tous deux pour plus proche correspond un cluster k' d'une base B et que $F(k_1, k') \geq 1$ et $F(k_2, k') \geq 1$, alors k est la fusion de k_1 et k_2, et k_1 et k_2 sont la fission de k.

Cette méthode permet donc une analyse très fine des variations de structures entre deux ensembles de données. En particulier, ce type d'analyse est très utile pour comprendre les variations de la structure d'un flux de données, telles que les apparitions et disparitions de clusters, ainsi que les phénomènes de fusions, fissions ou dérives de concepts.

4.5 Sélection de modèle et Stabilité

Un des problèmes les plus importants en analyse de données non supervisée est de déterminer les valeurs les plus pertinentes des paramètres des algorithmes utilisés. C'est ce qu'on appelle le problème de la sélection de modèles.

La sélection de modèles pour le clustering est un problème difficile. Les raisons évidentes sont que, contrairement à la classification supervisée, il n'y a pas de "vérité intrinsèque" à comparer aux résultats obtenus qui nous permettent de sélectionner une valeur de paramètre plutôt qu'une autre. L'une des questions les plus importantes dans la pratique est de savoir comment déterminer le nombre de clusters que l'on souhaite obtenir. Diverses méthodes ont été suggérées dans la littérature ; aucune n'est totalement convaincante. En particulier, on utilise fréquemment des indices dits "internes" pour estimer la qualité des résultats obtenus en fonction de certaines propriétés que devrait avoir un "bon" cluster. En particulier, on cherche en général des clusters homogènes et bien distincts. De nombreux indices internes on été proposés pour quantifier ces propriétés (parmi les plus utilisés on trouve les indices de Davis-Bouldin [35] et de Calinski-Harabasz [25], mais il en existe de nombreux autres). Ces méthodes souffrent généralement du fait qu'il est nécessaire de définir ce qu'est une "bonne classification" avant de pouvoir attribuer des scores différents à différentes valeurs des paramètres.

Ces dernières années, une nouvelle méthode est de plus en plus populaire : sélectionner le nombre de clusters à partir de la stabilité de la classification [156]. Au lieu de définir

"qu'est-ce qu'une bonne classification", la philosophie de base est simplement que la "bonne" classification devrait être une structure "stable" de l'ensemble des données. Autrement dit, s'il est appliqué à plusieurs ensembles de données de même distribution sous-jacente ou de même processus de production, un algorithme de clustering devrait obtenir des résultats similaires. Dans cette logique, ce qui est important n'est pas de savoir à quoi ressemblent les clusters (cela est pris en charge par l'algorithme de classification). Ce qui importe est de savoir s'ils peuvent être construits d'une manière stable. En particulier, un algorithme qui prend en compte les fluctuations aléatoires dues à l'échantillonnage pour construire des clusters (autrement dit qui fait du sur-apprentissage) ne va pas être stable, alors qu'un algorithme qui tient compte de la structure de la distribution sous-jacente (c'est à dire un algorithme capable de généraliser ses résultats) va être très stable. Il s'agit d'une méthode élégante, car elle évite de définir ce qu'est une bonne classification. Il s'agit aussi d'un méta-principe qui peut s'appliquer à n'importe quel algorithme de classification.

La figure 4.18 figure présente un exemple d'utilisation de la stabilité pour la sélection de paramètres (tiré de [156]). Il s'agit d'une distribution de données comprenant quatre clusters sous-jacents (les cercles noirs) et des échantillons différents de cette distribution (représentés par des losanges). Si nous partitionnons cet ensemble de données en $K = 2$ groupes, il existe deux solutions raisonnables : une séparation horizontale et une verticale. Si un algorithme de classification est appliqué à plusieurs reprises sur différents échantillons de cette distribution, l'une ou l'autre des solutions peut être obtenue en fonction des échantillons. Évidemment, ces deux solutions sont très différentes l'une de l'autre, et les résultats de la classification sont instables. Des effets similaires ont lieu si nous commençons avec $K = 5$. Dans ce cas, nous devons nécessairement scinder un cluster existant en deux groupes. Selon l'échantillon, n'importe lequel des quatre clusters peut être scindé en deux. Là encore, la solution de la classification est instable. Enfin, si l'on applique l'algorithme avec le nombre correct $K = 4$, on observe des résultats stables (non représentés sur la figure) : l'algorithme de classification découvre toujours les clusters corrects (sauf peut-être quelques outliers). Dans cet exemple, le principe de la stabilité nous permet de détecter le nombre correct de clusters.

Dans la section 2 nous avons utilisé des mesures de stabilité comme indice de qualité pour tester nos algorithmes. Ici nous souhaitons utiliser la stabilité comme outil de sélection de modèle. En particulier l'objectif est de déterminer la topologie optimale (nombre de neurones et dimensions de la carte) à utiliser dans DS2L-SOM, en fonction de la base de données étudiée. De plus nous souhaitons utiliser la mesure de similarité proposée dans la section 4.4 comme base nouvelle pour calculer la stabilité de nos algorithmes.

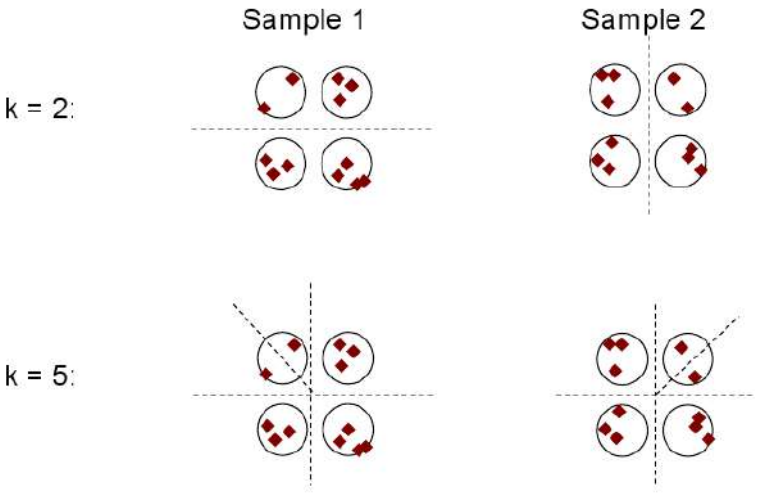


FIGURE 4.18 – La classification est instable si le nombre de clusters est trop petit (ligne 1) ou trop grand (ligne 2).

4.5.1 La stabilité

Dans cette partie nous analysons la stabilité de la classification d'un point de vue statistique. Une *classification d'un ensemble de données* $S = x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ est une fonction qui attribue des étiquettes $C_K : S \to 1, \ldots, K$ à tous les points de S. Ici, K désigne le nombre de clusters. Un *algorithme de classification* est une procédure qui prend un ensemble S de points en entrée et retourne en sortie une classification de S. La plupart des algorithmes de classification prennent un paramètre supplémentaire en entrée, à savoir le nombre K de clusters qu'ils sont censés construire. L'ensemble de données S est supposé être constitué de N points de données $x^{(1)}, \ldots, x^{(N)}$ qui ont été tirés indépendamment à partir d'une distribution sous-jacente inconnue P d'un espace χ. L'objectif final est d'utiliser ces points d'échantillonnage pour construire une bonne classification de l'espace χ sous-jacent.

Soit deux classifications C et C' de deux ensembles de données, il est possible de définir une distance d entre C et C'. Pour une distribution de probabilité P fixée, un nombre fixe K de clusters, et une taille de l'échantillon N fixée, l'*instabilité d'un algorithme de classification* est définie comme l'espérance de la distance entre deux classifications $C_K(S_N)$ et $C_K(S'_N)$ sur les différentes séries de données S_N et S'_N de taille N :

$$Instab(K, N) := E\left(d(C_K(S_N), C_K(S'_N))\right)$$

Dans la pratique, une grande variété de méthodes ont été mises au point pour calculer les scores de stabilité et les utiliser pour la sélection de modèle. Sur un plan très général, ils fonctionnent comme suit :

Soit S un ensemble de données et A un algorithme de classification qui prend en entrée le nombre de clusters k voulu.

1) Pour un nombre de groupe $k \in \{2 \dots k_{max}\}$

   1.1) Générer des versions perturbées $S_b, (b = 1, \dots, b_{max})$ des données originales.

   1.2) Pour chaque $b \in 1, \dots, b_{max}$, segmenter l'ensemble S_b avec l'algorithme A en k clusters de façon à obtenir une classification C_b.

   1.3) Pour chaque paire b et $b' \in 1, \dots, b_{max}$, calculer la distance $d(C_b, C_{b'})$ entre les deux classifications.

   1.4) Calculer l'indice d'instabilité :

$$\widehat{Instab}(k, N) = \frac{1}{b_{max}^2} \sum_{b,b'=1}^{b_{max}} d(C_b, C_{b'})$$

2) Choisir le paramètre k qui minimise l'instabilité :

$$K := \arg\min_k \widehat{Instab}(k, N)$$

Ce schéma donne un aperçu très rapide d'utilisation de la stabilité pour la sélection de modèle. Dans la pratique, de nombreux détails doivent être pris en compte.

Génération de versions perturbées du jeu de données. Pour être en mesure d'évaluer la stabilité d'un algorithme de classification donné, nous avons besoin d'exécuter l'algorithme à plusieurs reprises sur des ensembles de données légèrement différents. Pour cela, nous avons besoin de générer des versions perturbées de l'ensemble de données initial. En pratique, les méthodes suivantes ont été utilisées :

- Tirer un sous-échantillon aléatoire sans remise de l'ensemble de données originales (par exemple [95, 12, 53, 91]).
- Ajouter un bruit aléatoire sur les données d'origine [17, 109].
- Si l'ensemble de données d'origine est de grande dimension : utiliser différentes projections aléatoires dans des espaces de faible dimension, et ensuite classifier les ensembles de données de faible dimension [141].
- Si nous travaillons sur un modèle, générer les données de l'échantillon à partir du modèle [84].

– Générer un échantillon aléatoire des données originales par tirage aléatoire avec remise [156]. Cette approche évite le problème de la fixation de la taille du sous-échantillon. Ce type d'échantillonnage est la norme dans la littérature sur le «bootstrap» et pourrait également présenter des avantages dans le cadre de la stabilité.

Dans tous les cas, il y a un compromis qui doit être traité avec soin. Si les modifications des données sont trop importantes (par exemple, le sous-échantillon est trop petit, ou le bruit est trop important), alors la structure que nous voulons découvrir par la classification pourrait être détruite. Cependant, si les modifications sont trop faibles, alors l'algorithme de classification obtiendra toujours les mêmes résultats, et nous observerons une stabilité triviale. Il est difficile de quantifier ce compromis dans la pratique.

Quelles classifications comparer ? Différents protocoles sont utilisés pour comparer les classifications sur les différentes séries de données S_b.

– Comparer la classification de la base de données d'origine avec la classification obtenue sur les sous échantillons [95].
– Comparer les classifications des points de données communs aux deux sous-échantillons, après avoir calculé la classification de chaque sous-échantillon [12].
– Comparer les classifications de sous-échantillons disjoints [53, 91]. Ici, nous devons d'abord appliquer un opérateur pour étendre chaque classification dans le domaine de l'autre.

Distances entre classifications. Si deux classifications sont définies sur les mêmes points de données, il est alors facile de calculer un score de distance entre ces regroupements, avec des indices classiques tels que l'indice de Rand, l'indice de Jaccard, la distance de Hamming, etc ... [107]. Toutes ces mesures comptent, d'une manière ou d'une autre, les points ou les paires de points sur lesquels les deux classifications sont en accord ou en désaccord.

Scores de stabilité et leur normalisation. La méthode de mesure de stabilité décrite ci-dessus aboutit à un ensemble $d(C_b, C_{b'})_{b,b'=1,\dots,b_{max}}$ de valeurs de distances. Dans la plupart des approches, ces valeurs sont résumées par leurs moyennes :

$$\widehat{Instab}(k, N) = \frac{1}{b_{max}^2} \sum_{b,b'=1}^{b_{max}} d(C_b, C_{b'})$$

Cependant, il faut noter que $\widehat{Instab}(k, N)$ augmente avec k, quelle que soit la distribution sous-jacente des données. Voir par exemple la courbe en haut à gauche de la

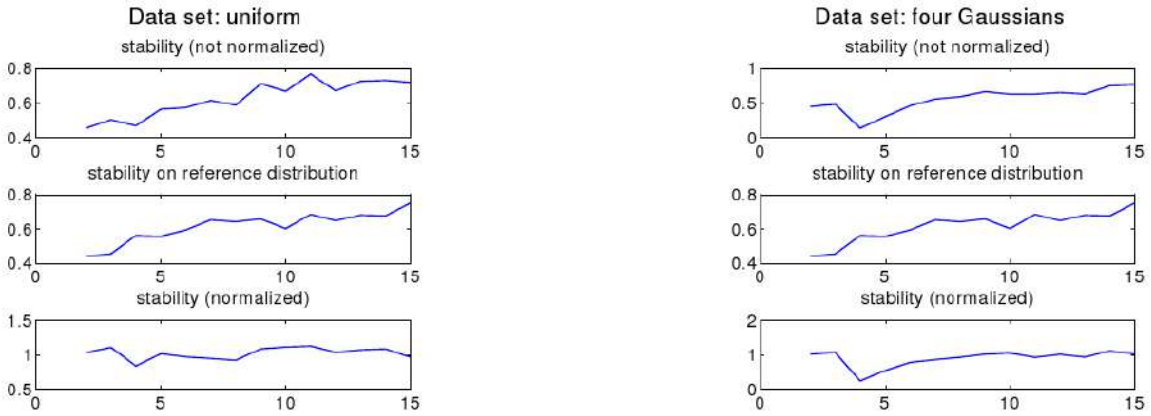


FIGURE 4.19 – Score de stabilité normalisé [156].

figure 4.19. L'utilisation de la stabilité pour la sélection de modèle nécessite donc une *normalisation* :

- Normalisation utilisant une distribution nulle de référence [53, 15]. Des échantillons sont générés répétitivement à partir d'une distribution nulle de référence. Cette distribution est définie dans le même domaine que les données d'origine, mais sans structure de classes. Un cas simple consiste à utiliser une distribution uniforme dans le domaine des données. Une solution plus pratique consiste à mélanger chaque dimension des données existantes et d'utiliser ces données mélangées comme distribution nulle. Chaque échantillon de distribution nulle est classifié par l'algorithme, puis un score de stabilité est calculé sur ces échantillons : $\widehat{Instab}_{null}$. La stabilité normalisée est alors définie comme $\widehat{Instab}_{norm} := \widehat{Instab}/\widehat{Instab}_{null}$.
- Normalisation par étiquetage aléatoire [91]. Pour commencer, chaque base de données S_b est classifiée comme décrit précédemment, pour obtenir les classifications C_b. Ensuite, les étiquettes de classes des données sont permutées aléatoirement. Pour finir, un score de stabilité est calculé sur les classifications initiales et un autre score est calculé sur les classifications correspondant aux données aux étiquettes permutées. La *stabilité normalisée* est alors définie comme $\widehat{Instab}_{norm} := \widehat{Instab}/\widehat{Instab}_{perm}$.

Une fois que les scores de stabilité normalisée ont été calculés, il est possible de choisir le nombre de clusters qui minimise l'instabilité normalisée :

$$K = \underset{k=2,\ldots,k_{max}}{\arg\min}\ \widehat{Instab}_{norm}(k, N)$$

Cette approche a été mise en application par exemple dans [12, 91].

4.5.2 Approche proposée et protocole expérimental

L'objectif de ce travail est de déterminer les dimensions optimales de la carte pour le clustering d'une base de données avec DS2L-SOM. Pour cela nous souhaitons calculer la stabilité de l'algorithme en fonction des dimensions choisies pour la carte. De plus, nous proposons d'utiliser une mesure de similarité entre segmentations basée sur la dissimilarité CBd de la section 4.4. Autrement dit nous allons comparer deux partitions en comparant les distributions de densité des clusters obtenus dans les deux cas. La mesure proposée pour comparer deux partitions P_1 et P_2 est la suivante :

1) Pour chaque cluster k appartenant à une des bases, trouver le cluster k' le plus similaire dans l'autre base selon la dissimilarité CBd :

$$k' = \arg\min_i CBd(k, i)$$

2) Soit N_k le nombre de neurones dans le cluster k , la dissimilarité $Diss$ entre P_1 et P_2 se calcule comme suit :

$$Diss(P_1, P_2) = \frac{\sum_{k \in P_1 \cup P_2} N_k CBd(k, k')}{\sum_{k \in P_1 \cup P_2} N_k}$$

L'estimation de la stabilité d'un algorithme et la sélection du meilleur paramètre pour une base de donnée S se calcule alors selon le protocole suivant :

1) Pour chaque nombre de neurones $Nn \in \{Nn_{min}, \ldots, Nn_{max}\}$ faire :

   1.1) Déterminer les dimensions optimales de la carte $L * l$ en fonction de Nn et des données à analyser comme dans [151] :

$$L/l = \lambda_1(S)/\lambda_2(S) \text{ et } L + l = Nn$$

   λ_1 et λ_2 sont les deux premières valeurs propres d'une ACP effectuée sur les données. Ces valeurs donnent une idée des proportions de l'ensemble des données. Il faut aussi noter que les valeurs de L et l sont arrondies à l'entier le plus proche et qu'il est possible d'obtenir les mêmes dimensions pour des choix initiaux différents du nombre de neurones. Une solution alternative, qui n'a pas été testée ici, consisterait à choisir toutes les dimensions possibles pour chaque valeur de N_n, mais cela augmenterait énormément le nombre de tests à effectuer.

1.2) Générer des versions perturbées $S_b, (b = 1, \ldots, b_{max})$ des données originales par tirage aléatoire avec remise (Bootstrap).

1.3) Pour chaque $b \in 1, \ldots, b_{max}$, segmenter l'ensemble S_b en k clusters de façon à obtenir une partition P_b.

1.4) Pour chaque paire b et $b' \in 1, \ldots, b_{max}$, calculer la distance $Diss(P_b, P_{b'})$ entre les deux partitions.

1.5) Pour chaque $b \in 1, \ldots, b_{max}$ permuter aléatoirement les valeurs des données, indépendamment pour chaque variable, de façon à obtenir une base de données de distribution nulle b_{null}.

1.6) Pour chaque paire b_{null} et b'_{null}, calculer la distance $Diss(P_{b_{null}}, P_{b'_{null}})$ entre les deux partitions.

2) Calculer l'indice d'instabilité normalisé :

$$\widehat{Instab_{norm}}(Nn) = \frac{\sum_{b,b'=1}^{b_{max}} Diss(P_b, P_{b'})}{\sum_{b,b'=1}^{b_{max}} Diss(P_{b_{null}}, P_{b'_{null}})}$$

3) Choisir le paramètre Nn_{opt} qui minimise l'instabilité normalisée :

$$Nn_{opt} := \arg\min_{Nn} \widehat{Instab_{norm}}(Nn)$$

Pour tester ce protocole, nous avons utilisé une dizaine de bases de données réelles et artificielles, de dimensions comprises entre 2 et 13. Les bases “Lsun”, “Wingnut”, “Engytime”, “Target”, “TwoDiamonds”, “Hepta”, “Tetra” et “Chainlink” font partie du Fundamental Clustering Problem Suite (FCPS, [149]) et sont décrites dans la subsection 2.2. Les données “Iris” et “Wine” sont des bases de données réelles bien connues de dimension 4 et 13 respectivement (UCI Repository, [49]). Nous faisons varier le nombre de neurones de $2\sqrt{(}N)$ à $8\sqrt{(}N)$, N étant le nombre de données. 25 versions perturbées des données ont été testées pour chaque valeur de Nn.

Pour chacune de ces bases, une segmentation “attendue” est connue. Nous allons donc tester la validité de la sélection automatique des dimensions de la carte vis-à-vis de la segmentation attendue. Pour cela, nous calculons une valeur moyenne de l'indice de Jaccard [77] sur un ensemble de perturbations par bootstrap de la base de données pour chaque topologie testée. Ainsi, pour chaque nombre de neurones utilisés, nous avons un indice $Jacc(Nn)$ qui reflète la validité des résultats obtenus avec cette topologie. Nous pouvons donc choisir le nombre optimal de neurones Nn^* à utiliser avec DS2L-SOM pour l'obtention du résultat attendu :

$$Nn^* = \arg\max_{Nn} Jacc(Nn)$$

.

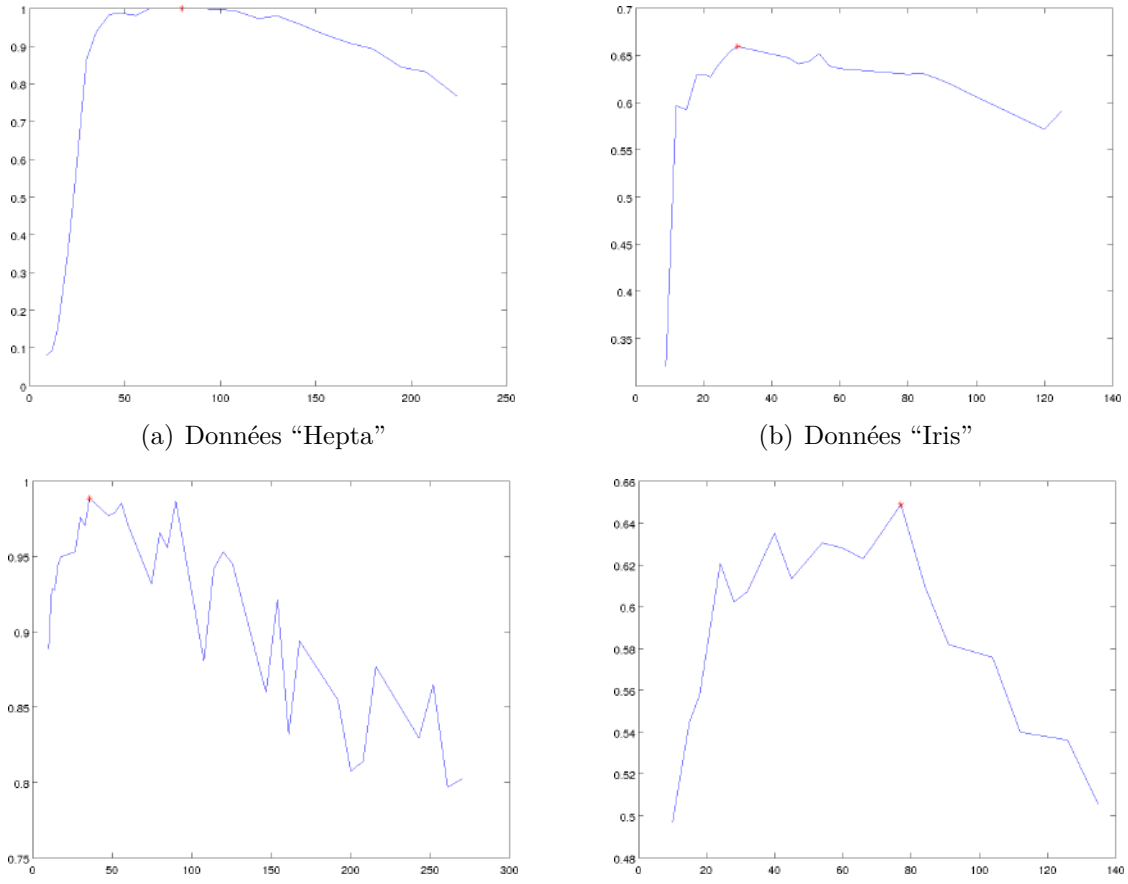
Si la valeur de Nn_{opt} obtenue en maximisant la stabilité est pertinente, les résultats obtenus avec Nn_{opt} doivent être proches des résultats optimaux obtenus avec Nn^*. Autrement dit, $Jacc(Nn_{opt})$ doit être proche de $Jacc(Nn^*)$. Nous avons donc calculé pour chaque base de données un indice de la qualité de Nn_{opt} :

$$Q(Nn_{opt}) = \frac{Jacc(Nn_{opt})}{Jacc(Nn^*)}$$

4.5.3 Résultats

Pour commencer nous avons voulu vérifier l'influence du nombre de neurones sur les performances de DS2L-SOM. Pour cela il suffit de regarder l'évolution de $Jacc(Nn)$ en fonction de Nn. La figure 4.20 montre quelques exemples. Comme on le voit, il est important pour des résultats optimums de ne pas choisir un nombre de neurones trop grand ou trop petit. Cependant, DS2L-SOM est capable d'obtenir de bons résultats pour un intervalle relativement grand de nombre de neurones. Par exemple les résultats obtenus pour "Wine" ou "Iris" sont quasiment aussi bons avec 20 qu'avec 80 neurones (soit 4 fois plus). De même, les cartes de 40 à 130 neurones donnent des résultats très proches pour "Hepta". Une autre remarque importante est que lorsque les groupes ne sont pas bien séparés, non seulement le nombre de neurones mais aussi le rapport longueur/largeur de la carte sont importants. Par exemple pour "Wine" et, de manière plus prononcée, pour "TwoDiamonds", on obtient de nombreux optimums locaux qui dépendent du rapport L/l. En effet, "TwoDiamonds" une carte de dimensions $14 * 4$ (56 neurones, $L/l = 3,5$) va donner des résultats équivalents qu'une carte $18 * 5$ (90 neurones, $L/l = 3,6$), alors qu'une carte intermédiaire de dimensions $15*5$ (75 neurones, $L/l = 3$) va donner des résultats nettement moins bons. Il faut donc tenir compte à la fois du nombre de neurones et des dimensions de la carte pour obtenir des résultats optimaux.

La Table 4.2 donne les valeurs de $Q(Nn_{opt})$ pour chaque base de données. Plus cette valeur est proche de 1, plus les résultats obtenus avec le nombre de neurones recommandé par le test de stabilité sont bons. Plus elle est proche de 0, moins les recommandations du test de stabilité sont pertinentes. Ces valeurs sont comparées aux valeurs $Q(Nn_{jacc})$ obtenues par une mesure de stabilité classique qui se base sur le même protocole que celui décrit plus haut mais qui utilise une mesure de Jaccard au lieu de CBd pour comparer deux partitionnements. Pour comparer les deux partitions avec l'indice de Jaccard, les deux bases de données perturbées sont fusionnées puis



(a) Données “Hepta” (b) Données “Iris”

(c) Données “TwoDiamonds” (d) Données “Wine”

FIGURE 4.20 – Visualisations des valeurs moyennes de l’indice de Jaccard des résultats de DS2L-SOM sur données perturbées pour differentes valeurs du nombre de neurones. L’optimum est noté en rouge.

segmentées en fonction des deux segmentations obtenues pour les SOM. La comparaison se fait sur les deux segmentations obtenues. Ces valeurs sont aussi comparées aux valeurs $Q(Nn_{heu})$ obtenues en utilisant l’heuristique proposée par [151] dans la SOM-Toolbox ($Nn_{heu} = 5\sqrt{(N)}$, N étant le nombre de données).

Les expérimentations montrent que les résultats obtenus avec une topologie sélectionnée selon notre mesure de stabilité sont très proche de l’optimal (moins de 10% d’écart dans la grande majorité des cas, moins de 1% d’écart pour la moitié des bases de données testées). Les résultats obtenus sont similaires à ceux obtenus selon une mesure de stabilité classique utilisant l’indice de Jaccard. Cependant, pour ces données, notre mesure donne des résultats supérieurs ou égaux à ceux de la mesure classique pour 7 des 10 bases testées. La comparaison des distributions des clusters semble donc

TABLE 4.2 – Valeurs de $Q(Nn_{opt})$, $Q(Nn_{jacc})$ et $Q(Nn_{heu})$ pour chaque base de données (en pourcentage).

Bases de données	$Q(Nn_{opt})$	$Q(Nn_{jacc})$	$Q(Nn_{heu})$
Chainlink	100	100	96
Engytime	81	99	93
Hepta	99	98	97
Iris	95	98	96
Lsun	100	98	94
Target	91	95	94
Tetra	90	83	95
TwoDiamonds	99	99	87
Wine	94	90	100
Wingnut	100	96	96

une alternative pertinente aux mesures de stabilité classiques. On peut noter aussi que l'heuristique très simple proposée par [151] donne des résultats en moyenne aussi bons que la sélection selon la stabilité, bien qu'elle donne des meilleurs résultats que les mesures de stabilité uniquement dans 2 cas sur les 10 testés. Ainsi, bien que les mesures de stabilités paraissent efficaces pour la sélection des dimensions de la carte à utiliser, le choix heuristique reste une méthode très avantageuse en temps de calcul et qui mérite d'être utilisé.

Nous avons donc testé dans ce travail différentes méthodes pour déterminer les dimensions optimales de la carte à utiliser dans DS2L-SOM. Nous avons montré que ce choix est important pour obtenir de bons résultats. On peut noter que le choix du ratio longueur/largeur de la carte est aussi important que choix du nombre de neurones dans ce cas. Cependant DS2L-SOM est capable de bonnes performances pour une gamme relativement large de topologies de la carte, en particulier du fait de la phase de fusion qui joue un rôle stabilisateur sur les partitions obtenues.

Trois méthodes on été testées : une méthode heuristique, une méthode classique de mesure de stabilité et une nouvelle mesure de stabilité basée sur une comparaison des densités des différents clusters. Des tests sur des bases de données artificielles et réelles ont montré la pertinence des trois approches, bien qu'aucune ne soit nettement plus efficace que les deux autres. Dans ce contexte, la méthode heuristique semble donc un choix particulièrement intéressant, puisqu'elle évite d'utiliser des tests de stabilité, particulièrement coûteux en temps de calcul.

4.6 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode de modélisation de la structure des données, basée sur l'apprentissage d'une SOM, ainsi qu'une mesure de dissimilarité entre modèles. Les avantages de cette méthode sont, d'une part, une grande rapidité de calcul (mise à jour « en ligne » des estimations) et une faible quantité d'informations à stocker pour chaque modèle, mais aussi une grande précision dans la modélisation obtenue. Les résultats obtenus sur des bases de données artificielles et réelles valident la pertinence de l'approche.

Ces algorithmes peuvent être combinés avec les algorithmes de classification présentés dans le chapitre 2 pour une analyse efficace de données spatio-temporelles. Nous présentons dans le chapitre suivant deux applications réelles de ce type d'analyse, ainsi qu'une proposition de méthode pour l'analyse et la classification des flux de données.

Chapitre 5

Analyse de données évolutives

5.1 Introduction

L'analyse des données spatio-temporelles, telles que les données de trajectométrie ou les flux de données, font partie des applications importantes dans de nombreux domaines.

Dans ce chapitre, nous présentons deux applications réelles d'analyse de trajectoires obtenue avec la technologie RFID (Radio Frequency IDentification). La première application est une étude biologique visant à analyser la dynamique du comportement collectif d'une colonie de fourmis tropicales lors d'un déménagement [1]. La seconde application est une analyse du déplacement de clients dans un grand magasin pour la découverte des grandes zones de fréquentation et des fréquences de passages dans les différentes allées [2].

La troisième section de ce chapitre est une proposition de méthode d'analyse de grands flux de données, permettant de stocker les propriétés du flux et de comparer différentes périodes pour détecter d'éventuelles variation temporelles. Cette tâche est réalisée en calculant un modèle de la structure du flux à différentes périodes et en mesurant des similarités entre ces structures.

L'ensemble de ces analyses reposent sur les algorithmes présentés dans les chapitres précédents.

5.2 Analyse du comportement dynamique d'une colonie de fourmis en déplacement

L'apprentissage non-supervisé est un outil très important pour la détection automatique de sous-groupes pertinents dans un ensemble de données, lorsqu'on n'a pas d'information à priori sur la structure interne de ces données. Ce type de méthode est donc particulièrement adapté à la fouille de données issues d'études expérimentales, pour lesquels on a en général peu d'information a priori.

DS2L-SOM est un outil efficace de classification qui permet de découvrir et de représenter simplement une quantité d'information importante sur la structure des données. Nous avons donc souhaité appliquer cette méthode à des données issues de la recherche

1. ANR Sillages N° 05 BLAN 017701
2. ANR CADI N° 07 TLOG 003

expérimentale, de façon à montrer son efficacité pour l'extraction de connaissances dans ce domaine et la découverte de résultats scientifiques.

La RFID (Radio Frequency IDentification) est une technologie avancée d'enregistrement de données spatio-temporelles de traçabilité. Les Tags RFID, constitués d'une puce et d'une antenne, peuvent être détectés par un lecteur RFID capable d'enregistrer la présence d'un grand nombre de Tags en un seul scan et de les identifier. Un ordinateur est utilisé pour stocker dans une base de données les positions de chaque Tag lors de chaque scan, ce qui autorise un grand nombre d'analyses possibles après enregistrement. Grâce à la miniaturisation, la RFID offre l'avantage de l'automatisation et échappe aux contraintes du suivi vidéo.

Les systèmes RFID peuvent être utilisés pour étudier les sociétés animales. Les sociétés animales sont des systèmes dynamiques complexes caractérisés par beaucoup d'interactions entre les individus. Une telle structure dynamique provient de la synergie de ces interactions, les capacités individuelles dans le traitement de l'information et la diversité de réponses individuelles [38, 51, 42, 134]. Le but de ce travail est de développer un nouveau système autonome basé sur la RFID pour suivre l'activité spatio-temporelle des groupes et de développer de nouveaux outils pour le traitement automatique de ce type de données. Une version miniaturisé du système RFID peut être adaptée pour être utilisée en condition naturelle et connaît déjà des applications en éthologie et des utilisations en élevage. Cependant, l'expérimentation à partir de système RFID génère de grandes bases de données nécessitant des méthodes d'analyse adaptées afin d'en comprendre le sens, de déceler des relations entre événements et d'en déduire des modèles de comportement. Ces objectifs font de ce travail un projet interdisciplinaire combinant l'informatique avec les sciences du comportement et des systèmes complexes.

Le cadre applicatif choisi pour cette étude est la division du travail au sein d'une colonie de fourmis, en collaboration avec le professeur Dominique Fresneau et Jerzy Witwinowski du laboratoire d'éthologie de l'université Paris 13 [3]. Un dispositifs RFID miniaturisé à été développé pour ce modèle biologiques par la société *SpaceCode* [4]. Basé sur des produits commercialisés, il nécessite peu de développement. Il s'agit d'un réseau d'antennes de lecteurs dans un espace contraint notamment à des points de passages obligés dans des nids artificiels. Ces antennes sont connectées à un détecteur qui routera les informations vers un ordinateur. La technologie RFID n'a pas d'effets observables sur le comportement des fourmis [125] et permet de suivre les patterns d'activité d'un grand nombre d'individus sur de grandes périodes temporelles.

3. http://www-leec.univ-paris13.fr/
4. http://www.spacecode-rfid.com/

L'évolution de ces données au cours du temps et leur position spatiale nécessitent d'explorer simultanément plusieurs ensembles de données multi-variables. En statistique, des méthodes d'analyse factorielle multiple et de classification ont été développées ces dernières années, mais elles sous-estiment le plus souvent le caractère spatial des données. Par contre, dans le domaine de l'apprentissage numérique (en particulier connexionniste), le traitement des données spatio-temporelles est un problème assez "jeune" et représente une voie de recherche très prometteuse ouvrant d'excellentes perspectives.

5.2.1 Suivi par RFID d'une colonie de fourmis

L'objectif de cette partie est l'analyse de l'occupation spatiale des individus de la colonie de fourmis en fonction de leurs rôles sociaux. En effet, dans une colonie de fourmis, chaque individu joue un rôle particulier pour la survie du groupe. C'est ce qu'on appelle la division du travail. Ce phénomène est commun à tous les insectes sociaux, dans une mesure plus ou moins importante. Chez les fourmis les rôles sociaux les plus fréquents sont : recherche de nourriture, soin du couvain (œufs, larves, cocons) et de la reine, défense du nid, nettoyage du nid, etc ...

L'organisation de la colonie autour de ces différents rôles est encore peu connu. Nous nous occupons ici de l'organisation spatiale des individus, en utilisant notre méthode de classification pour regrouper automatiquement des individus présentant des comportements similaires d'occupation des salles. L'objectif étant de découvrir une éventuelle spécialisation de chaque individu pour l'occupation de certaines salles du nids, puis de mettre en rapport cette spécialisation avec le rôle social de chaque fourmi.

5.2.1.1 Description des données et de la méthode utilisée

Le dispositif expérimental est une fourmilière artificielle composée de trois salles (N1, N2 et N3) et d'une zone de récolte (ME) reliées linéairement par trois tunnels (Figure 5.1). Ces tunnels sont équipés de deux détecteurs RFID qui détectent le passage et la direction des individus portant un tag RFID lorsqu'ils changent de salle. La reine (non taggée) se trouve le plus souvent dans la salle N3, le plus loin de la zone de récolte.

L'espèce étudiée ici, *Pachycondyla tarsata*, établit des colonies monogynes pouvant aller jusqu'à 2500 ouvrières monomorphes [71]. Elle construisent des nids souterrains avec des salles inter connectées d'une surface de $1200m^2$. Un tunnel de $130m$ entre plusieurs salles a même été observé [23]. La périphérie des nids constitue des positions

FIGURE 5.1 – Dispositif RFID expérimental

avancées qui renforcent le réseau. La reine et la couvain sont particulièrement mobile dans le nids afin d'optimiser l'élevage du couvain. Une colonie de 33 fourmis a été suivie en continu dans le dispositif au Laboratoire d'Ecologie Expérimentale et Comparée (LEEC, Paris13) pendant 36 jours, chaque individu portant sur le thorax un tag RFID (Figure 5.2). Ce tag consiste en une puce électronique reliée à une antenne, l'ensemble pesant moins de 40mg (soit environ 25% du poids d'une fourmi).

FIGURE 5.2 – Fourmi avec tag RFID

Chaque individu a été codé sous forme vectorielle par la proportion de temps passé dans chaque zone, puis nous avons appliqué S2L-SOM sur ces données.

5.2.1.2 Résultats

La figure 5.3 représente la carte obtenues avec DS2L-SOM et les différentes étapes de la correction selon la densité. Chaque hexagone est le représentant d'un ensemble

d'individus très similaires. Plus les représentants de deux individus sont proches sur la carte, plus les individus sont similaires.

FIGURE 5.3 – Classification par DS2L-SOM avec correction selon la densité à partir des données RFID.

La segmentation finale de la carte met à jour quatre types d'individus en ce qui concerne l'occupation des salles (Figure 5.4). On peut caractériser chaque type en fonction du profil de leurs représentants (Figure 5.4 à droite).

Ainsi les individus «bruns» sont caractérisés par une occupation importante de la salle N3 (salle de la reine), à l'exclusion des autres. Au contraire, les fourmis « bleues » et «jaunes» passent plus de temps que les autres dans la zone de récolte, mais alors que les «jaunes» passent beaucoup de temps (relativement aux autres) dans la salle N1, les « bleues » restent plus longtemps dan la salle N2. Enfin les individus du type «orange» présentent un profil intermédiaire, ils ne se caractérisent pas par une occupation particulière des salles par rapport aux autres fourmis.

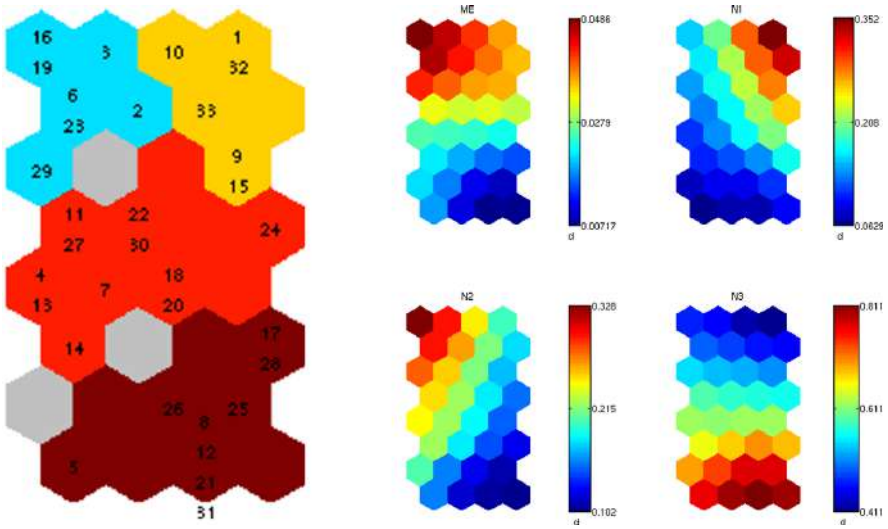


FIGURE 5.4 – Carte obtenue à partir des données RFID et profil des représentants.

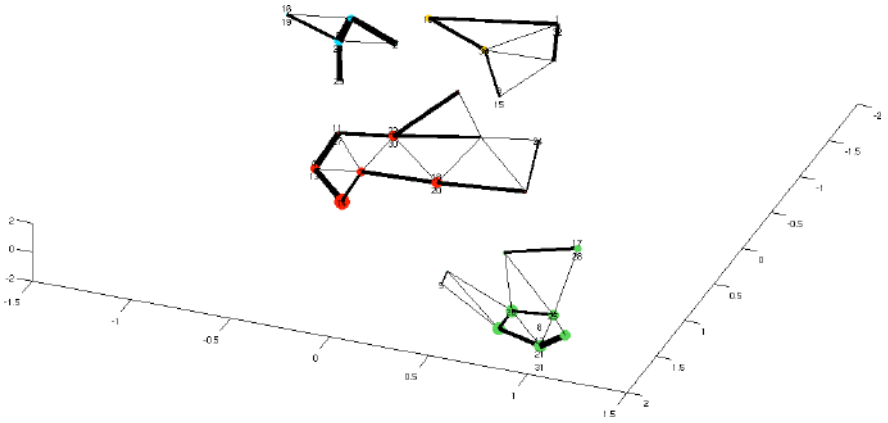


FIGURE 5.5 – Projection de Sammon des prototypes et leurs connexions, à partir des données RFID.

La représentation selon une projection de Sammon (Figure 5.5) permet une analyse plus fine de la structure de chaque groupe et de leurs relations. Le groupe « brun » est composé d'un noyau de forte densité d'individus (8, 12, 21, 31, 26, 25) très représentatifs de leur groupe (c'est à dire associés à des prototypes bien connectés entre eux) et d'un ensemble d'individus (3, 17 et 28) marginaux par rapport à ce groupe (associés à des prototypes peu connectés aux autres). Les prototypes du noyau sont très proches les uns des autres et particulièrement éloignés des autres groupes. Cela signifie que les individus représentés sont similaires entre eux et fortement spécialisés dans leur occupation de l'espace (ici la salle N3, salle de la reine). Les individus marginaux ont un comportement intermédiaire avec les autres groupes, ils sont moins spécialisés que les autres membre du groupe « brun ». Les individus du groupe « bleu » sont, eux aussi,

bien spécialisés dans leur occupation de l'espace, c'est à dire la salle N2 et la zone de récolte. Leurs prototypes sont proches et bien connectés entre eux et, en particulier, les individus 6 et 23 sont très représentatifs du groupe. Par contre, les individus du groupe «jaune » présentent un comportement représentatif plus diversifié, leurs prototypes étant plus éloignés les uns des autres, tout en étant biens connectés. Pour finir, le groupe « orange » est caractéristique d'un comportement généraliste. La plupart des prototypes sont bien connectés aux autres et il n'y a pas de comportements marginaux (sauf l'individu 24), mais on note aussi une distance importante entre ces prototypes, signe d'une grande variété de comportements au sein des individus du groupe.

Cette segmentation est sans doute l'expression d'une répartition des tâches entre les individus de la colonie. Les fourmis des groupes « bleus » pourraient être spécialisées dans la récolte et le traitement de la nourriture. Les fourmis « jaunes », qui passent moins de temps à l'extérieur et ont un comportement spatial plus diversifié pourraient s'occuper des tâches d'entretien de la fourmilière alors que les fourmis du groupe «brun» s'occuperaient de la reine et du couvain. Le groupe «orange» pourrait être composé d'individus peu spécialisés ayant un rôle plus polyvalent (ou bien pas de rôle particulier) dans la colonie. Toutes ces hypothèses on été vérifiées et validées expérimentalement par la suite par les biologistes du projet.

Les nouvelles techniques de classification non supervisée (DS2L-SOM) permettent donc de traiter efficacement des données issues d'études expérimentales. Elles permettent, de plus, de visualiser de façon simple et très efficace les résultats obtenus. Ici nous avons pu mettre en évidence les caractéristiques de l'organisation dans l'espace d'un groupe d'individus. Nous avons regroupé les individus selon des catégories de comportements caractéristiques et nous avons pu décrire de façon détaillée ces comportements. Cela nous a permis d'émettre des hypothèses (pouvant être vérifiées expérimentalement) sur l'organisation des tâches au sein du groupe.

5.2.2 Structure des déplacement lors d'un déménagement

Dans une deuxième étape, une colonie de fourmis a été observée après la création d'un changement artificiel de climat, ce qui a provoqué le déplacement de la colonie vers un autre nid. Les migrations de colonies sont des évènements dangereux car chaque membre de la colonie – particulièrement la reine et le couvain – devient vulnérable à la prédation ou court le risque de se perdre. Par conséquent, la synchronisation relative de la migration de la reine et du transport du couvain sont des enjeux stratégiques pour la colonie car elle peut réduire sensiblement le risque encouru par ces membres essentiels.

Par exemple, le déplacement de la reine et le transport du couvain ne se produisent pas au hasard pendant la délocalisation du nid, ils se produisent au contraire au milieu de la séquence de migration. De plus, les différentes catégories de couvain sont transportés lors d'une séquence spécifique (les cocons d'abord, puis les larves, finalement les œufs, [120]). Ces résultats correspondent à des stratégies robustes à l'échelle de la colonie car ils ne sont pas liés à la distance de migration, ni à la taille de la colonie ou du couvain.

Nous avons suivi et analysé le mouvement d'une colonie de fourmis *Pachycondyla tarsata* composée de fourmis de grande taille facile à marquer et à suivre. Les fourmis *Ponerine* sont une espèce tropicale dont les colonies sont composées d'une centaine à un millier d'individus [118]. Dans ces espèces, les reines ne sont souvent pas très différentes des ouvrières, mais il existe une grande variété de structures sociale, allant de la monogamie à la polygamie, avec certaines espèces qui n'ont pas de caste de reine [119, 110]. Dans la nature, les fourrageuses solitaires cherchent des termites par l'orientation visuelle [70]. Elles sont aussi capables d'utiliser un recrutement massif à travers une piste de phéromones lorsqu'elles découvrent une source importante de nourriture [71] et le transport des proies est réparti entre les différentes fourrageuses, chacune faisant seulement une partie du voyage retour. Les ouvrières récoltent aussi des restes d'insectes et contribuent au renouvellement des ressources écologiques [36]. C'est par conséquent un modèle intéressant pour l'étude en laboratoire des déménagements de colonies.

5.2.2.1 Dispositif expérimental

La dynamique de la répartition des tâches en fonction de l'ontogenèse des ouvrières a été largement décrite dans des études théoriques précédentes. Cependant, le manque d'outils adéquats pour suivre le comportement des fourmis n'a pas permis une validation expérimentale des hypothèses théoriques.

Dans cette section, nous avons pour objectif d'étudier la dynamique d'un déménagement d'une colonie de fourmis. Nous souhaitons caractériser et analyser l'évolution de l'ensemble de la colonie, comprendre comment les fourmis agissent et répartissent leur rôle pendant le processus du déménagement. La cinétique qui caractérise la séquence de départ du nid d'origine, le passage par la zone de fourragement (l'extérieur) et l'arrivée dans le nouveau nid seront analysés. Ces actions nécessitent une grande coordination chez les ouvrières. Nous nous attendons e à ce que les fourrageuses commencent l'exploration du nouveau nid et mettent en place le recrutement pour mener plus de fourmis peu mobiles (nurses et fourmis inactives) vers le nouveau nid. Les fourmis qui transportent le couvain, et plus important, celles qui s'occupent du déplacement de la reine

vers le nouveau nid sont inconnues. De même, nous ne savons pas si la distinction établie des rôles sociaux en situation stable conditionnera le rôle assumé par un individu spécifique pendant la phase de migration.

Nous avons donc suivi le déménagement d'une colonie de 55 ouvrières à l'intérieur d'un dispositif RFID (environ 4 heures de suivi).

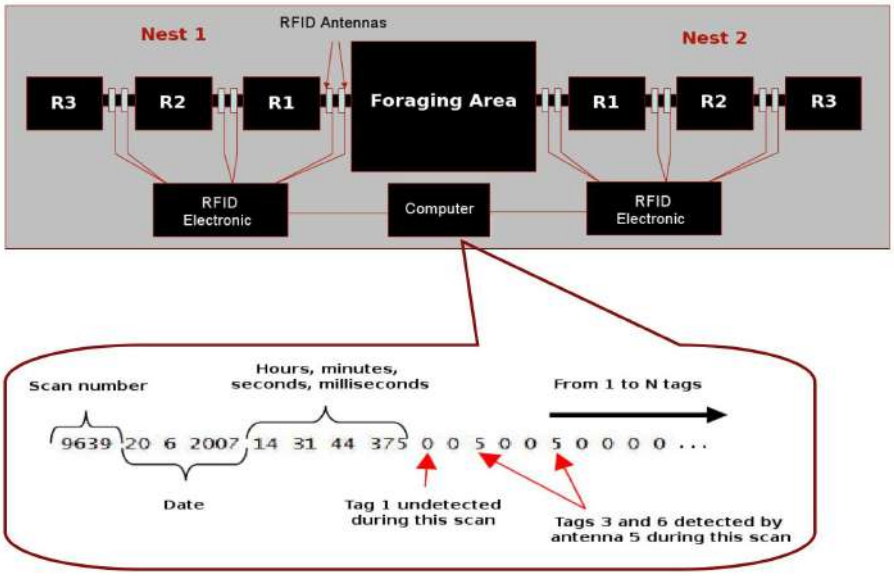


FIGURE 5.6 – Le dispositif RFID expérimental et un exemple de détection enregistrée.

Le dispositif expérimental pour cette expérience est composé de trois salles et d'une zone de fourragement connectées linéairement par six tunnels (Figure 5.6). Au début de l'expérience, la reine (non marquée) et le couvain (œufs, larves et cocon) sont situés dans la salle 3 du premier nid, le plus éloigné de la zone de fourragement. Chaque tunnel est équipé de lecteurs RFID (numérotés de 1 à 12 de la salle 3 dans le nid 2 à la salle 3 dans le nid 1) qui détectent le passage et la direction des individus taggés entre les salles. La position d'un individu peut être déduite sans ambiguïté par l'information fournie par les lecteurs dans les tunnels. Le manque de détection implique que l'individu est à l'extérieur du tunnel et donc dans une des sept salles. Le lieu exact d'un tag (donc d'un individu) peut être déduit de la direction du passage dans un tunel. L'information enregistrée par les lecteurs est traitée par un matériel électronique RFID et envoyée ensuite à l'ordinateur qui crée et stocke les fichiers de données.

Le système effectue régulièrement un "scan" de tous les tags présents dans les tunnels (environ trois scans par seconde). Les fichiers de données sont au format texte. Ils indiquent, pour chaque scan, le numéro du scan, la date et l'heure et, pour chaque tag (donc pour chaque individu), quelle antenne est activée (Figure 5.6). Si, pendant

un scan, aucun individu n'est détecté, rien n'apparait dans le fichier de données. Un simple traitement de ces fichiers rend possible l'obtention de l'information spatiale pour chaque individu.

En raison des limites imposées par la miniaturisation, les tags RFID peuvent parfois ne pas être perçus par les appareils de détection. Un taux de détection manquantes allant de 5 à 15% a été observé.

Au temps $t = 0$ nous ouvrons le premier nid et allumons une lampe à néon produisant une lumière puissante (répulsive pour les fourmis), puis nous enregistrons le déplacement de la colonie jusqu'à ce que la totalité du couvain ait été déplacée dans le second nid (environ 4 heures).

5.2.2.2 Segmentation des trajectoires

Pré-traitement des données : Nous utilisons le fichier d'enregistrement RFID pour calculer la séquence de déplacement individuelle de chaque fourmi. Cette séquence est une fonction qui donne l'emplacement de la fourmi à n'importe quel moment pendant le déplacement (voir Figure 5.7 pour un exemple). L'emplacement est codé par un numéro de lecteur si la fourmi est sous celui-ci et un nombre réel pour représenter la salle entre deux lecteurs (par exemple si la fourmi est dans la salle 2 du Nid 2, entre les lecteurs 2 et 3, l'emplacement sera 2,5).

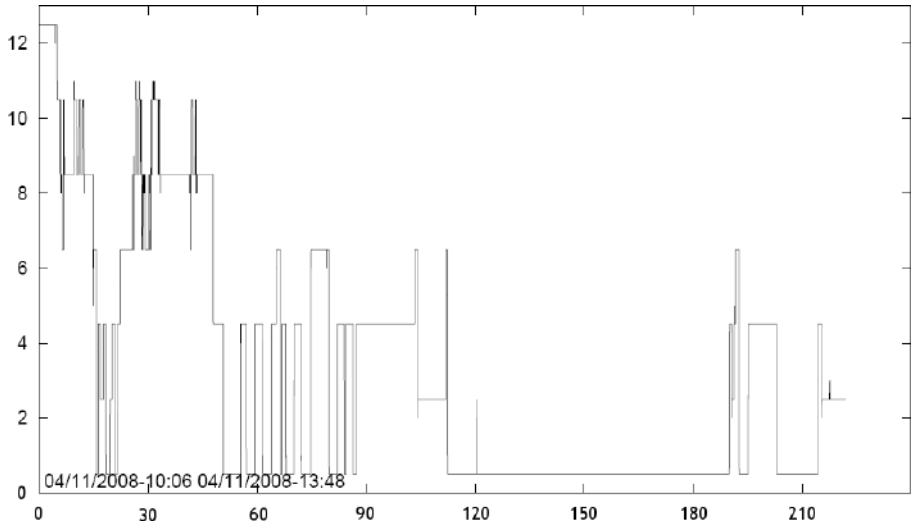


FIGURE 5.7 – Exemple d'une séquence de déplacement. Le temps depuis le début de l'expérience (en minutes) est en abscisse. La position de la fourmi est en ordonnée.

Cependant, ce que nous voudrions analyser est la variation du comportement spatial d'une fourmi au fil du temps. Pour ce faire, un comportement spatial courant doit être défini. Ici, nous ne pouvons pas juste choisir l'emplacement courant, parce que de cette manière nous perdrions toutes les informations dynamiques telles que « la fourmi se déplace rapidement » ou « la fourmi fait des aller-retours entre deux salles ». Par conséquent, nous choisissons de définir le comportement courant par le temps passé dans chaque emplacement (information statique) et le nombre de sorties de chaque emplacement (information dynamique) pendant un laps de temps de 10 minutes centré sur le temps courant (Figure 5.8).

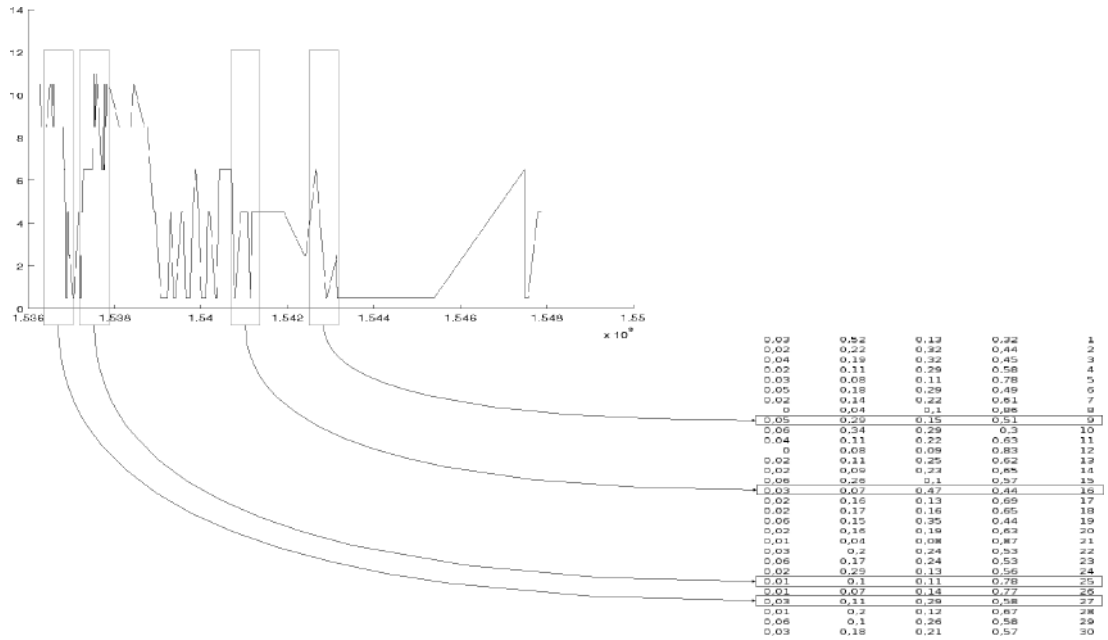


FIGURE 5.8 – Exemple de représentation d'un comportement spatial courant (seules 4 variables sont montrées ici).

Évidemment, cette définition implique quelques corrélations entre les descriptions de deux comportements courants s'ils sont séparés par moins de 10 minutes, puisque les deux fenêtres temporelles se chevauchent. Cela nous permettra de détecter les changements rapides de comportement. Comme il existe 19 emplacements dans le dispositif RFID (7 salles et 12 lecteurs), chaque fenêtre temporelle est codée sous forme vectorielle de 38 variables normalisées (une variable statique et une variable dynamique pour chaque emplacement).

Détection de sous-séquences homogènes individuelles : Dans le but de regrouper des comportements courants identiques et de détecter les changements de

comportement au fil du temps, nous appliquons l'algorithme DS2L-SOM sur les fenêtres temporelles de chaque séquence individuelle. La mesure de la distance utilisée par l'algorithme pour cette étude est la distance Euclidienne. La Figure 5.9 donne un exemple des résultats obtenus avec DS2L-SOM pour une fourmi.

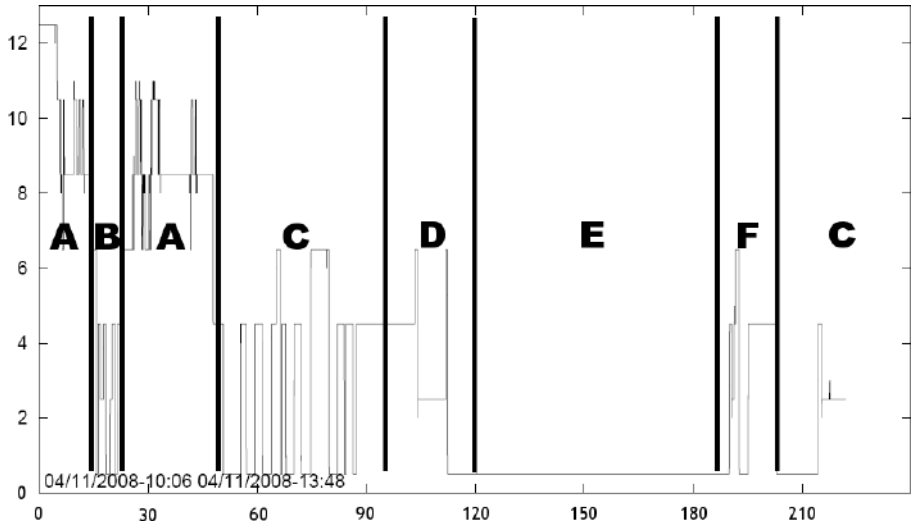


FIGURE 5.9 – Exemple d'une détection automatique de sous-séquences homogènes pour un individu.

Lorsque la lumière s'allume, cette fourmi commence à se déplacer rapidement à l'intérieur du Nid 1 (emplacement 7 et plus) avec quelque sorties dans la zone de fourragement (emplacement 6.5). Ce comportement homogène a été trouvé automatiquement par DS2L-SOM, c'est un ensemble de comportements courants regroupés en un cluster (appelé A par l'algorithme), nous l'appellerons « comportement A ». Après 15 minutes environ, cette fourmi se déplace du Nid 1 vers le Nid 2 puis se déplace rapidement à l'intérieur du Nid 2. Cela a été détecté comme un comportement homogène (appelé B). Par la suite, la fourmi revient dans le Nid 1 et montre un comportement très semblable au premier. DS2L-SOM détecte automatiquement que la fourmi exprime de nouveau le même comportement et lui donne le même nom. Ensuite, la fourmi revient dans le Nid 2 mais ne se déplace pas aussi rapidement que le dans le comportement B, ce qui constitue alors un autre comportement (appelé C). Enfin, la fourmi reste dans le Nid 2 jusqu'à la fin de l'expérience, exprimant différents comportements (ayant des caractéristiques statiques et dynamiques variables).

Pour finir, nous avons demandé à des experts du domaine d'étiqueter chaque sous séquences des individus, de façon à pouvoir comparer les trajectoires entre elles. Cet étiquetage manuel n'est pas optimal, nous proposons cependant une méthode automatique par la suite.

5.2.2.3 Model probabiliste des déplacements

Cette section à été réalisée en collaboration avec le professeur Attilio Giordana et Ugo Gallassi de l'université de Turin en Italie.

Le dispositif RFID fournit seulement une observation spatiale des individus. Aucune information n'est fournie concernant ce qui se passe à l'intérieur de la salle, mais seule la durée de stationnement d'une fourmi à l'intérieur d'une salle peut être connue. Par ailleurs, les capteurs ne sont pas fiables ayant un taux de détection manquant allant de 5 à 15%.

L'objectif est de reconstruire l'évolution de l'activité d'un individu dans le contexte de l'environnement social, sous la pression d'un événement dangereux. Plus particulièrement, nous voulons découvrir quelles sortes d'activités sont exprimées pendant la phase de migration, combien d'individus sont en charge de chaque activité, et quand est-ce qu'un individu change d'activité.

Atteindre cet objectif requière la résolution des problèmes suivants :

(i) Reconstituer les trajectoires les plus probables prisent par les fourmis en considérant que de nombreux passages dans les tunnels ne sont pas observés à cause d'un manque de détection.

(ii) Caractériser les différents schémas d'activités.

(iii) En déduire des modèles d'activités.

(iv) Segmenter et étiqueter les trajectoires en fonction de l'activité qui a le plus probablement produit la séquence d'actions observées.

L'outil de modélisation dont nous avons besoin doit avoir de bonnes capacités de traitement des états partiellement observables et doit être capable de modéliser les durées de stationnement des individus dans les différentes zones. À cette fin, l'approche modèle graphique [111] semble la plus prometteuse.

La théorie des probabilités offre un cadre pour la modélisation de l'évolution des processus caractérisés par des phénomènes aléatoires inhérents, ou opérants dans des environnements trop complexes pour une analyse précise. L'idée centrale est que les lois statistiques qui régissent l'évolution d'un système peuvent être estimées à partir d'un ensemble d'apprentissage de traces décrivant son histoire passée. Cette approche nous permet de déduire un modèle à partir d'un groupe relativement restreint de séquences.

En particulier, deux outils émergent en tant que bons candidats pour la segmentation et l'étiquetage des séquences en présence d'états cachés : les Modèles de Markov Cachés ou HMM [Hidden Markov Model, 121] et les Conditional Random Fields [CRF 90]. Des résultats récents [111] sont en faveur de la CRF, qui est souvent plus performante que les HMM. Néanmoins, l'exigence de la modélisation de la durée nous suggère d'adopter l'approche HMM. En fait, les graphes orientés sont un choix naturel pour modéliser des successions d'événements qui se caractérisent par une causalité temporelle. Des méthodes performantes d'extension des HMM pour modéliser des durées sont disponibles, alors que les CRF ont été peu étudiés dans ce sens [112].

L'outil choisi pour cette étude est une variante des HMM appelé S-HMM [Structured HMM, 57], qui offre des fonctionnalités spécifiques pour la modélisation des stationnements dans les salles. Un S-HMM est un graphe orienté constitué, selon les règles de composition précise, de plusieurs sous-graphes indépendants (des blocs), structurés selon le paradigme utilisé dans la Programmation Orientée Objet. Un bloc est constitué d'un ensemble d'états, seuls deux d'entre eux (l'état *initial* I et l'état *final* E) sont autorisés à être connectés à d'autres blocs. Les blocs peuvent être imbriqués les uns l'intérieur des autres. L'hypothèse de base qui sous-tend une modélisation par S-HMM est qu'une séquence d'observations $O = o_1, o_2, o_3, ..., o_T$ peut être segmentée en une série de sous-séquences $O_1, O_2, ..., O_N$, chacune générée par un sous-processus ayant seulement des interactions faibles avec ses voisins [20]. Cette hypothèse est réaliste dans de nombreuses applications pratiques, comme, par exemple, la reconnaissance vocale [121, 122] ou l'analyse d'ADN [41].

Modélisation de l'activité des fourmis : Un modèle de l'activité des fourmis doit attribuer une distribution de probabilité sur l'ensemble de toutes les trajectoires possibles pour une fourmi accomplissant une activité particulière. Soit s une séquence d'observations. En comparant les différentes probabilités assignées à s par un ensemble de différents modèles d'activités, il est possible d'en déduire l'activité qui a le plus probablement généré s.

L'observation d'un chemin est une séquence de paires $\langle t_i, s_i \rangle$ enregistrés par les capteurs RFID, avec t_i le moment de la détection en msecs et s_i l'ID du capteur ayant détecté la présence de la fourmi à ce moment. Dans un S-HMM on considère un temps discret. De ce fait, une transformation de la représentation numérique des capteurs en une représentation discrète (symbolique) a été définie, qui conserve la précision implicite du codage original. Les séquences symboliques sont encodées en utilisant un alphabet $\mathbf{A} = \{A, B, C, D, E, F, G, H, I, J, K, L, .\}$, où les lettres de A à L correspondent aux lecteurs RFID de 1 à 12, respectivement, et " . " désigne un intervalle de temps pen-

dant lequel l'individu n'a pas été détecté. La transformation d'une séquence numérique en séquence symbolique est obtenue en divisant le temps en intervalles discrets d'une *seconde*. Chaque seconde, si un lecteur RFID a détecté l'individu, le symbole correspondant est ajouté dans la séquence symbolique ; sinon on ajoute un " . ". Ainsi, après transformation, le stationnement dans une salle est représenté comme une chaîne de " . ". En outre, un passage non détecté dans un tunnel sera aussi rapporté comme un " . ".

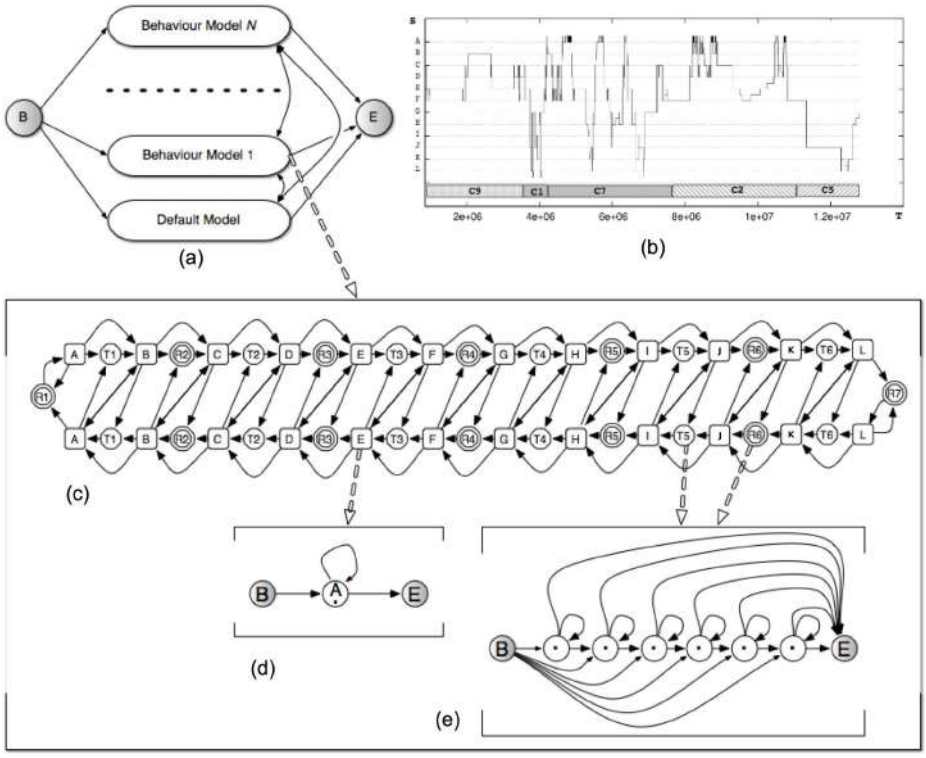


FIGURE 5.10 – Le HMM Structuré modélisant le comportement (c'est-à-dire les activités) des fourmis. (a) L'étiqueteur d'activité : un S-HMM à trois niveaux utilisé pour la modélisation du comportement de la colonie. (b) Un exemple de séquence étiquetée obtenue en utilisant l'étiqueteur. (c) Un modèle indépendant à deux niveaux a été appris pour chaque activité. (b) Un bloc de base codant le passage sous un capteur. (c) Un bloc de base modélisant la durée de stationnement dans une chambre ou un tunnel.

Après avoir expérimenté différentes architectures pour le modèle d'une activitée, celle visible sur la figure 5.10(c) a été choisie. Il s'agit d'un S-HMM à deux niveaux : le niveau supérieur modélise les trajectoires à travers l'environnement, tandis que le niveau inférieur modélise les observations détectées par les lecteurs RFID et la durée de stationnement dans les salles et les tunnels. Les états du niveau supérieur définissent une

double chaîne reliée aux extrémités. Les états représentés par un seul cercle représentent un stationnement dans un tunnel, tandis que ceux représentés par un double cercle représentent un stationnement dans les salles du nid ou dans l'aire de fourragement. Les états désignés avec des carrés représentent des capteurs. Chaque état est associé à un bloc de niveau inférieur, qui modélise la distribution de probabilité pour le stationnement dans l'emplacement correspondant ou le processus de génération des émissions observables des capteurs. Sur la figure 5.10(c), la chaîne du haut modélise les déplacements de l'ancien nid au nouveau nid, tandis que la chaîne du bas modélise les déplacements allant en direction de l'ancien nid. Les changements de directions provoquent le passage d'une chaîne à l'autre.

Apprentissage du modèle : À partir de l'architecture décrite dans la Figure 5.10(c), les modèles des différentes activités ont été estimés afin de construire un *étiqueteur d'activité*. Celui-ci est utilisé pour déduire la trajectoire la plus probable d'une fourmi et de segmenter et étiqueter cette trajectoire en fonction de l'activité qui a le plus probablement généré les signaux détectés par les lecteurs RFID. Pour finir, le comportement global de la colonie pendant le déménagement a été reconstruite à partir des trajectoires étiquetées.

La procédure complète d'apprentissage de l'étiqueteur d'activité combine des algorithmes d'exploration de données et l'intervention manuelle d'un expert du domaine. L'expert du domaine est très bon pour la détection des changements d'activité dans les séquences comportementales, mais il est peu performants dans des tâches nécessitant l'analyse systématique de grandes quantités de données. D'autre part, les algorithmes d'apprentissage sont performants pour la découverte de régularités et de similitudes dans les séquences. De cette collaboration, les groupes d'activités caractéristiques sont progressivement individualisés et modélisés. La procédure comporte les étapes suivantes, qui sont répétées jusqu'à convergence vers des modèles stables :

Soit $\mathcal{L}$ l'ensemble des séquences à étiqueter et L_i un sous-ensemble de $\mathcal{L}$, utilisé pour l'itération i.

1) Étiquetage des séquences avec la version actuelle de l'étiqueteur.
2) Validation et correction des étiquettes par les experts.
3) Segmentation de chaque sequence en fonction des étiquettes associées.
4) Regroupement des segments ayant la même étiquette.
5) Estimation d'un modèle λ_k pour chaque groupe C_k.

6) Construction d'un nouvel étiqueteur à partir des modèles appris lors de l'étape précédente, éventuellement cet étiqueteur peut être entraîné par l'algorithme Baum-Welch.
7) Ajout dans L_i de nouvelles sequences extraites de $\mathcal{L}$ pour obtenir un nouvel ensemble d'apprentissage L_{i+1}.

À la fin de la procédure, toutes les séquences de $\mathcal{L}$ sont étiquetées par l'étiqueteur construit lors de la dernière étape. Un exemple de séquence étiquetée est présenté sur la Figure 5.10(b).

Il faut noter que lors de la première itération on n'a pas encore d'étiqueteur. Nous avons donc utilisé la segmentation des séquences présentées en 5.2.2.2. Les différents segments ont été étiquetés par les experts du domaine pour obtenir une base d'apprentissage initiale pour la modélisation probabiliste.

De cette façon, nous avons obtenu une description très fine des différentes activités exprimées lors du déménagement et de la dynamique de ces activités.

5.2.2.4 Résultats

A la fin du processus d'apprentissage, huit groupes de comportements apparaissent $A_i (1 \leq i \leq 8)$. Un dernier groupe a été défini (A0), il correspond aux segments d'activité ne correspondant pas à un pattern homogène. Les traces d'activité de chaque fourmi ont été étiquetées en fonction de ces comportements. A partir des séquences étiquetées, des paramètres globaux ont été extraits, traçant le profil de la colonie de fourmis pendant la phase de migration. La Figure 5.11 montre l'évolution du paramètre N_i correspondant au nombre d'individus impliqués dans l'activité $A_0, \ldots, A_8$.

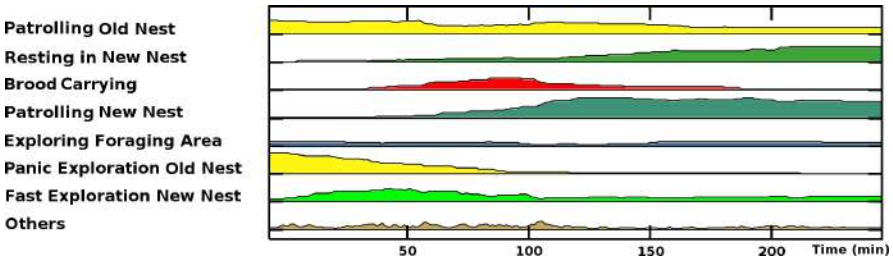


FIGURE 5.11 – Evolution du nombre d'individus impliqués dans les différentes activités.

Ces résultats ont été utilisés par les éthologistes pour formuler l'hypothèse d'une description plausible de chaque comportement.

A0 ***Patterns non structurés.*** Les fourmis ne participent à aucune activité et leur chemin à travers les salles n'est pas structuré. Une fourmi peut rester longtemps dans une salle, changer ensuite de salle en fonction d'évènements aléatoires : interaction sociale (parfois agressive), bruit dans l'environnement, danger, soif, faim, etc...

A1 ***Exploration rapide du nouveau nid.*** Les fourmis viennent de découvrir le nouveau nid et commencent une exploration active et rapide du nouveau site. Elles se déplacent rapidement à travers toutes les salles afin d'évaluer la salubrité du nouveau site et de décider rapidement si la migration doit commencer vers ce nouveau nid.

A2 ***Mouvements de panique.*** Ce comportement est le plus souvent exprimé par les nurses : à cause des nouvelles conditions environnementales (mauvaise lumière et augmentation de la température) l'ancien nid n'est plus approprié pour un bon soin du couvain. Les fourmis commencent alors à se déplacer rapidement à l'intérieur du vieux nid afin de trouver une salle avec les conditions appropriées. Une telle salle n'existant plus dans l'ancien nid, ces fourmis sont alors prêtes à migrer.

A3 ***Mouvements de panique dans l'ancien nid.*** Ce comportement est très semblable à A2, mais les fourmis sont plus craintives (ce comportement semble être surtout l'expression des plus jeunes fourmis). Par rapport au comportement A2, les fourmis restent plus proches du couvain et de la reine et ne sortent jamais.

A4 ***Patrouilles générales.*** C'est une exploration générale de l'environnement. Ce comportement peut être exprimé par quelques fourrageuses qui ne sont pas réellement impliquées dans l'activité de la migration, ou ce peut être le début d'un comportement de protection comme cela a été observé chez certaines autres espèces. Dans ce cas, les fourmis se déplacent à travers les deux nids et la zone de fourragement afin d'identifier les problèmes potentiels (prédateurs, zones dangereuses...).

A5 ***Transport.*** Ces fourmis transportent quelque chose : la reine, un cocon, une larve ou un œuf. Ce comportement est caractérisé par de nombreux aller-retours réguliers entre les deux nids. Il s'agit d'un pattern très caractéristique. Parfois il peut être exprimé par les fourmis qui ne transportent rien mais qui agissent comme des transporteuses (elles suivent les transporteuses ou se déplacent jusqu'à trouver quelque chose à transporter).

A6 ***Préparation du nouveau nid.*** Le nouveau nid est maintenant connu comme étant sûr : il n'y a pas de lumière, pas de perturbations et la plupart des fourmis y sont installées (la colonie est de nouveau bien structurée). Cependant, quelques ouvrières ont besoin de préparer le nid pour un soin optimal du couvain (trouver

de l'eau à l'extérieur afin d'augmenter l'humidité à l'intérieur du nouveau nid dans certaines salles, trouver des matériaux de construction, trouver de la nourriture, etc ...).

A7 ***Patrouilles dans l'ancien nid.*** Ce comportement est une patrouille défensive dans l'ancien nid en réaction à la perturbation. Les fourmis agissent pour défendre et protéger la reine et le couvain jusqu'à leur transfert.

A8 ***Patrouilles dans l'ancien nid et la zone de fourragement.*** Ce comportement est une patrouille défensive dans le nouvel environnement. L'objectif est probablement de défendre et de protéger la reine et le couvain pendant le déplacement et l'installation à l'intérieur du nouveau nid.

D'un point de vue éthologique, les résultats précédents sont d'une grande aide pour comprendre comment les tâches sont distribuées pendant le déménagement d'un nid. En fait, nous obtenons une description très précise de la dynamique de l'ensemble de la colonie pendant toutes les phases de la migration, ce qui nous permet d'émettre de fortes hypothèses au sujet de la fonction des différents comportements pendant la phase de déménagement du nid. Certains résultats concordent avec des travaux précédents, particulièrement en ce qui concerne les comportements qui peuvent être observés dans la zone de fourragement. Par exemple, la dynamique du comportement de transport détectée par le système correspond aux résultats présentés dans [120]. Ces hypothèses doivent maintenant être validées par la répétition de l'expérience avec différentes colonies et différentes espèces. Une compréhension complète du processus de migration basée sur des expérimentations systématiques serait un pas important à venir pour la recherche sur les insectes sociaux.

Bien que nous utilisions peu d'individus dans cette étude, la méthode est parfaitement adéquate pour l'étude de milliers d'individus, avec des comportements décrits par un grand nombre de paramètres spatio-temporels. L'exemple que nous avons développé ici peut être étendu à des situations multiples où le contrôle des capacités des individus aussi bien que leur intégration à l'échelle collective sont nécessaires, par exemple pour l'étude de régulations de flux dans les groupes sociaux (contrôle des foules, panique de masse, etc ...).

5.3 Analyse des déplacements des clients dans un grand magasin

Dans cette section, nous cherchons à étudier les activités spatio-temporelles des clients pendant leurs achats dans un grand magasin. Jusqu'à présent, peu de recherches ont été entreprises dans ce sens. Les questions habituelles sont les suivantes : Comment les clients se déplacent à travers le magasin, s'arrêtent-ils dans chaque secteur ou passent-ils d'un endroit à un autre d'une manière plus directe ? Suivent-ils une trajectoire unique ou une grande variété de trajectoires [92] ?

De telles données dynamiques sont difficiles à collecter et ce genre d'étude implique la disponibilité d'outils d'observation adéquats. Les technologies de traçabilité (RFID) remplissent parfaitement ce rôle, elles permettent en effet d'automatiser le suivi des individus en détectant à la fois leur localisation et leurs mouvements. Pour ce travail, nous avons utilisé un dispositif RFID basé sur des produits commercialisés. Il se compose d'un réseau de lecteurs RFID qui envoie les informations à un ordinateur. Ces lecteurs détectent les mouvements des étiquettes RFID collées sur le fond des paniers à la disposition des clients.

L'objectif applicatif de ce travail est l'exploration des données enregistrées par le système RFID pour l'analyse des comportements d'achat des clients dans le magasin. En particulier, nous souhaitons connaitre le temps moyen passé dans chaque secteur du magasin de façon à détecter les zones à forte et à faible fréquentation (points chauds et points froids). Nous voulons en outre analyser les patterns généraux des trajectoires des clients.

5.3.1 Dispositif de suivi RFID

Les mouvements des clients durant leurs achats ont été enregistrés à l'aide d'un dispositif RFID développé par *SpaceCode*. Pour ce faire, nous avons collé une étiquette RFID sur une vingtaine de paniers en plastique (Figure 5.12). Ces paniers sont mis à la disposition des clients pour leurs achats.

Le site utilisé pour ce travail est un magasin spécialisé dans la vente d'objets décoratifs ($6000m^2$). Il accueille en moyenne 2500 visiteurs chaque mois. Un dispositif RFID de 4 lecteurs à été installé dans différents secteurs du magasin pour l'enregistrement des trajectoires des clients. Les premiers tests sur le site ont montré que le positionne-

FIGURE 5.12 – Panier avec une étiquette RFID.

ment des lecteurs est très délicat, de nombreux tests on été réalisés pour trouver les emplacements optimums des lecteurs.

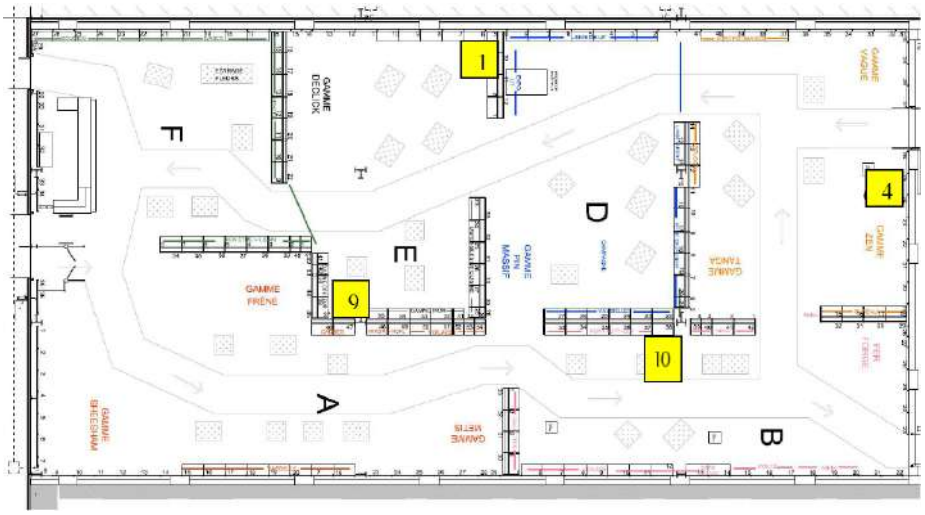


FIGURE 5.13 – Le dispositif RFID expérimental, les cases jaunes représentent la position des antennes dans le magasin.

La figure 5.13 montre la position finale des lecteurs dans le magasin. Il faut noter que les numéros des lecteurs (1, 4, 9 et 10) représentent le dernier numéro de leur adresse IP. Les informations enregistrées par les lecteurs sont collectées puis sont envoyées vers un ordinateur pour la création et le stockage de ces données.

5.3.2 Les données

Les fichiers de données indiquent, pour chaque balayage d'antenne (un balayage par seconde environ), le numéro de l'étiquette détectée, l'adresse IP du lecteur qui a détecté l'étiquette, ainsi que la date et l'heure (Figure 5.14). Lorsque aucune étiquette n'est détectée, rien n'apparaît dans le fichier des données .

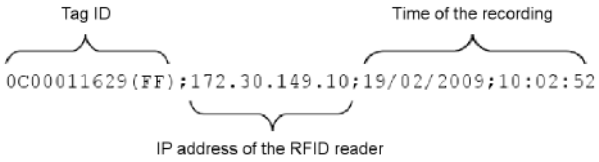


FIGURE 5.14 – Exemple d'enregistrement dans le fichier de données.

5.3.3 Pré-traitements des données

Lorsqu'un client se déplace dans le magasin, il est détecté successivement par les différentes antennes, selon sa localisation. Cependant, la plupart des secteurs du magasin sont couverts par plus d'une antenne. De ce fait, chaque panier est détecté le plus souvent par plus d'une antenne (cependant, les enregistrements des différentes antennes ne sont pas synchronisés). Ce recouvrement ajoute une information plus fine sur la position exacte du client, mais rend aussi beaucoup plus complexe la compréhension de la séquence de déplacement. En outre, les données enregistrées se révèlent être extrêmement bruitées, en particulier à cause de perturbations du signal RFID résultant de la présence de structures métalliques et des corps des clients (voir la Figure 5.15 pour un exemple de trajectoire enregistrée).

Pour pouvoir analyser les trajectoires des clients, nous devons inférer la position du client dans le magasin à chaque instant à partir du signal RFID complexe et bruité. Pour ce faire, nous définissons toutes les 10 secondes de parcours (instant t) une fenêtre temporelle de 2 minutes centrée sur t qui représente l'emplacement actuel du client. Chaque fenêtre temporelle est alors stockée sous la forme d'un vecteur représentant le nombre de fois où l'étiquette à été détectée par chacune des antennes pendant ces 2 minutes. Bien entendu, cette définition implique une certaine corrélation entre la description de deux fenêtres séparées par moins de 2 minutes, puisque les deux fenêtres se chevauchent. Cela nous permet de détecter quand un client se déplace d'un endroit à un autre, par la détection des changements soudains dans la description de l'emplacement actuel.

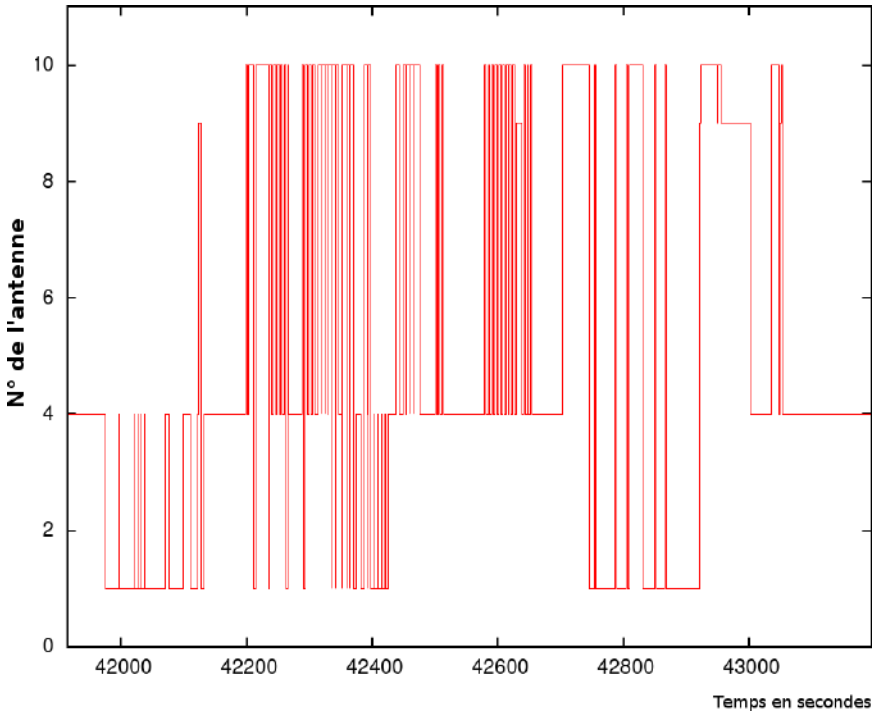


FIGURE 5.15 – Exemple d'une séquence de déplacement (∼20 min) représentée par le numéro de l'antenne détectant le panier en fonction du temps (secondes).

5.3.4 Détection de la position du client et des changements de secteurs

Afin de détecter les changements de secteur des client pendant leur parcours, nous appliquons un algorithme de clustering sur les vecteurs des fenêtres temporelles de chaque client. L'algorithme utilisé est DS2L-SOM qui calcule une SOM Enrichie sur l'ensemble des vecteurs fenêtres, puis utilise des informations de distance et de densité pour détecter les changements soudains dans les patterns de détection des antennes, de façon à regrouper les fenêtres similaires (voir Partie 2). La figure 5.16 donne un exemple de résultat obtenu pour la trajectoire d'un client.

Dans cet exemple, le client commence par traverser un secteur détecté par le lecteur 1 et parfois par le lecteur 4. Cette structure homogène a été détectée automatiquement par l'algorithme DS2L-SOM. Toutes les fenêtres temporelles correspondant à cet endroit sont regroupées en un seul cluster (appelée A par l'algorithme), nous l'appellerons "secteur A". Après environ 5 minutes, le client se déplace vers un autre secteur (appelé B) et ainsi de suite...

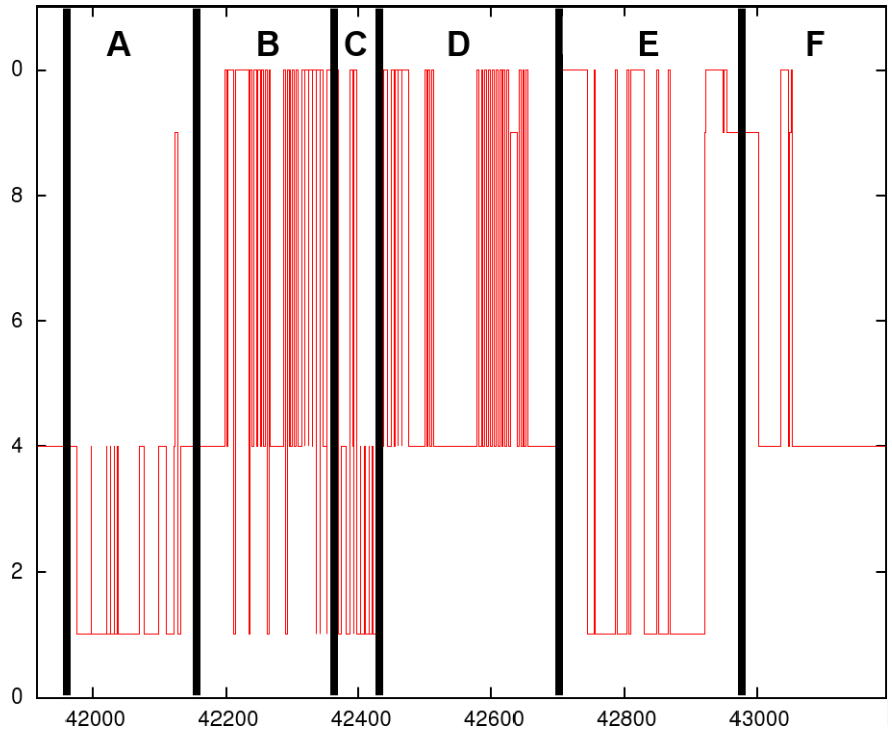


FIGURE 5.16 – Exemple de découpage automatique d'une trajectoire client.

5.3.5 Détection des patterns caractéristiques à l'échelle collective

La méthode utilisée à la section 5.3.4 nous permet de décrire la trajectoire de chaque client. Cependant, nous avons besoin maintenant d'une méthode pour comparer toutes ces séquences individuelles, de façon à pouvoir effectuer une analyse au niveau collectif (global). L'idée est de définir une mesure de (dis)similarité entre les déplacements élémentaires de chaque client, de façon à pouvoir comparer deux trajectoires entre elles. Pour ce faire, on calcule pour chaque cluster individuel (c-à-d pour chaque sous-séquence homogène) une fonction de densité représentative (cf. section 4.2). Pour finir nous avons défini une mesure de dissimilarité (cf. section 4.4) entre deux clusters C_K et C_L, représentés par deux séries de prototypes de SOM :

$$C_K = [\{w_i^K\}_{i=1}^{M^K}, f^K]$$

et

$$C_L = [\{w_i^L\}_{i=1}^{M^L}, f^L]$$

Avec M^K et M^L le nombre de prototypes w représentant C_K et C_L, et f^K et f^L les fonctions de densité associées respectivement à C_K et C_L.

Nous avons utilisé cette mesure pour calculer une matrice de dissimilarité avec tous les clusters de tous les clients enregistrés durant le premier jour de l'enregistrement. Puis nous avons utilisé une version modifiée de l'algorithme de classification adaptée aux matrices de dissimilarité (basée sur [32]). De cette façon, chaque "secteur" traversé par un client est associé aux "secteurs" similaires détectés chez les autres clients. Cela nous permet de comparer les différentes trajectoires entre elles (les clients passent-ils par les mêmes secteurs, l'ordre de passage est-il similaire, quel client reste le plus de temps dans tel secteur, etc...).

En nous basant sur une journée de suivi, nous avons détecté automatiquement avec cette méthode 6 "secteur" bien définis, chacun caractérisé par un pattern caractéristique de détection par les antennes. Il est maintenant possible d'utiliser la mesure de dissimilarité pour étiqueter chaque nouvelle séquence (c-à-d déterminer les "secteur" traversés) de chaque futur client. Cela est assez rapide pour être effectué en temps réel pendant la visite du client.

5.3.6 Résultats

La méthode d'analyse nous a permis de trouver, pour l'instant, 6 secteurs bien définis à partir de 4 antennes fonctionnelles. Cela signifie que nous sommes en mesure de définir plus de secteurs que le nombre de lecteurs (50% de plus), il s'agit d'une extraction d'information efficace. Les secteurs obtenus peuvent être décrits comme suit (voir aussi la Figure 5.17) :

- S1 : Couvert par l'antenne 9 seulement, elle correspond à l'entrée du magasin. Les paniers en attente de nouveaux clients sont détectés dans ce secteur.
- S2 : Couvert par l'antenne 1 seulement. Dans ce secteur les clients peuvent trouver des fleurs et des vases.
- S3 : Principalement couvert par les antennes 4 et 10. Dans ce secteur les clients peuvent trouver des objets en fer forgé.
- S4 : Principalement couvert par l'antenne 1, parfois par l'antenne 9 (meubles en bois).
- S5 : Principalement couvert par les antennes 9 et 10, parfois par les antennes 4 ou 1 (vaisselle et petits objets).
- S6 : Principalement couvert par les antennes 1 et 4, parfois par l'antenne 9 (Miroirs et linge).

La Figure 5.17 montre une estimation de l'emplacement des différents secteurs. Nous avons également calculé la fréquence de transition entre un secteur et un autre. Cela

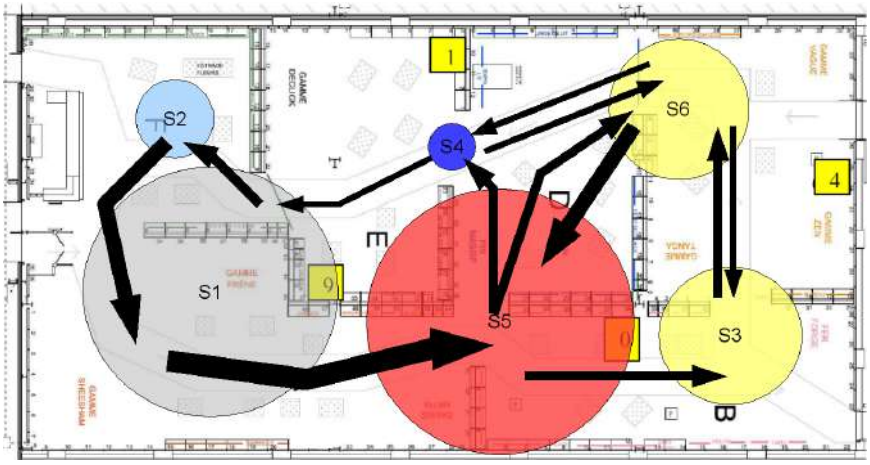


FIGURE 5.17 – Estimation de l'emplacement des différents secteurs. L'épaisseur des flèches est proportionnelle à la fréquence des transitions. Les fréquences les plus faibles n'ont pas été représentées. La taille de chaque secteur est proportionnelle au temps moyen passé à l'intérieur.

nous donne une idée de comment les clients circulent à l'intérieur du magasin. Nous pouvons constater par exemple que les clients prennent toujours le même itinéraire dans la première partie du magasin (S1 et S2), mais agissent plus librement au fond du magasin (S3 à S6). S5 semble être un secteur clé. On observe en effet de nombreuses transitions entre ce secteur et les secteurs voisins, de plus il est connecté à 4 des 5 autres secteurs.

Pour finir, nous avons calculé le temps moyen passé dans chaque secteur afin de trouver les points chauds et froids. Les résultats montrent que S5 est un endroit très fréquenté (48% du temps en moyenne) à la différence de S2 (6%) et S4 (2%) qui sont des secteurs où les clients ne s'arrêtent pas. If faut noter que nous n'utilisons pas S1 pour cette analyse, puisque les paniers en attente sont détectés dans ce secteur.

La méthode utilisée dans ce travail semble être un outil efficace de traitement des données pour les études utilisant un suivi RFID. En effet cette méthode est rapide, nécessite peu d'espace mémoire et est adaptée à un apprentissage en ligne. Nous avons ici mis à jour quelques caractéristiques de l'organisation spatiale des clients durant leurs achats dans un grand magasin. Il a été possible de traiter des données temporelles complexes et très bruitées. Nous avons aussi pu définir plus de secteurs bien définis que le nombre de lecteurs RFID dont nous disposions.

5.4 Analyse et classification des flux de données

Dans de nombreux cas, les bases de données sont en perpétuelle évolution, elles sont caractérisées par une structure variable dans le temps, de nouvelles données arrivant constamment. Parfois, l'évolution et la masse des données sont tellement importantes qu'il est impossible de la stocker dans une base et que seule une analyse « à la volée » est possible. Ces processus sont appelés « analyse des flux de données », ils ont fait l'objet de nombreux travaux ces dernières années du fait du nombre important d'applications possibles dans de nombreux domaines [73, 7, 114, 3, 61, 8]. Cependant, l'étude des flux de données est un problème difficile à cause des coûts de calculs et de stockages associés aux volumes mis en jeux. Dans le domaine de la fouille de données, les principaux enjeux pour l'étude des flux de données sont d'une part la description condensée des propriétés d'un flux [98, 59, 97] mais aussi la détection de variation ou de changement dans la structure du flux [131, 26, 1]. Parmi les nombreuses méthodes proposées pour résoudre ces problèmes, nous nous intéressons plus particulièrement aux méthodes basées sur un clustering des données.

La plupart des méthodes de classification adaptées aux flux de données supposent que la segmentation doit être calculée sur l'ensemble du flux (voir [114]). Cependant, le flux de données peut être vu comme un processus infini constitué de données qui évoluent constamment au cours du temps [3]. De ce fait, la segmentation des données du flux (autrement dit la structure du flux) est, elle aussi, en constante évolution. Ainsi le résultat d'une classification des données du flux pendant une période précise peut être très éloignée de la classification des données sur une autre période. Un algorithme de classification doit donc être capable de calculer la segmentation (ou la structure) des données du flux sur différentes périodes de temps, et être capable de comparer les structures des données sur ces différentes périodes. La détection de nouveauté dans la structure d'une base de données à été étudiée par de nombreux auteurs (Voir par exemple [94, 83] et plus particulièrement dans le domaine des flux de données [58, 2, 45]. Une difficulté supplémentaire vient du fait que le flux représente une masse énorme de données, qu'il est impossible de conserver en mémoire. Si l'on veut garder un historique de la structure des données au cours du temps, il faut pouvoir décrire cette structure sous une forme extrêmement condensée (en particulier sans avoir à mémoriser chaque donnée).

La détection de changement (ou de nouveauté) dans la structure du flux à été étudiée par de nombreux auteurs [58, 45, 94, 83]. Cependant, dans la plupart des cas, la détection doit être effectuée en temps réel, en comparant une nouvelle donnée à un modèle des données perçues jusqu'à maintenant. Ces méthodes ne permettent pas une

analyse flexible de la variation du flux à différentes échelles et à différentes périodes. La méthode proposée par [2] permet cette flexibilité, en calculant et comparant quelques informations statiques sur les données à différentes périodes, mais ne permet pas une analyse structurelle du flux.

Nous avons proposé dans la partie 4 une méthode de description condensée de la structure des données ainsi qu'une mesure heuristique de dissimilarité entre ces modèles permettant de détecter les variations temporelles de la structure du flux. L'avantage de cette méthode est la comparaison de structures par l'intermédiaire des modèles qui les décrivent, ce qui permet des comparaisons à n'importe quelle échelle temporelle sans surcharge de la mémoire de stockage. Ainsi il est possible de comparer la structure du flux à deux périodes potentiellement très éloignées dans le temps, puisque les modèles décrivant ces périodes peuvent être stockés en mémoire à très faible coût.

Nous souhaitons maintenant adapter une variante des cartes auto-organisatrices adaptée à l'étude d'un flux de données pour obtenir une méthode de clustering qui évolue avec les données. De cette façon, il suffira de stoker régulièrement la carte segmentée. Cette carte sera représentative de la structure des données du flux à un instant donné. La comparaison de cartes représentant différents instants permettra une analyse fine des variations de la structure des données.

Cette partie présente donc un protocole général de l'analyse de la structure d'un flux de donnée et détaille deux étapes importantes de ce processus : une classification adaptée aux flux de données et une méthode de compression permettant d'adapter la quantité d'informations à l'espace de stockage disponible.

5.4.1 Protocole général de l'analyse des flux de données

Le protocole proposé pour le suivi et l'analyse d'un flux de données est le suivant :

1) Utiliser un algorithme, basé sur l'apprentissage d'une SOM, capable de modéliser et de suivre la structure du flux en temps réel (Cf. section 5.4.2).
2) Enregistrer le modèle de la structure du flux à intervalle régulier. Le modèle consiste en une SOM enrichie et segmentée par l'algorithme précédent, accompagné d'une estimation de la densité des données dans chaque cluster (Cf. chapitre 4.2).
3) Compresser les modèles stockés au fur et à mesure que de nouveaux modèles sont enregistrés, pour ne pas dépasser la capacité limite de stockage (Cf. section 5.4.3).

4) Comparer deux modèles, correspondant à des époques différentes, grâce à la mesure de similarité entre modèles présentée dans la section 4.4.

Ce protocole permet à la fois de suivre et d'enregistrer des informations précises sur la structure d'un flux en temps réel. Les informations sont stockées au mieux, en tenant compte des contraintes des capacités de stockage, ce qui permet de détecter *a posteriori* des variations dans la structure du flux à différentes époques et à différentes échelles.

5.4.2 Un nouvel algorithme de clustering des flux de données

5.4.2.1 L'algorithme Growing Cellular Probabilistic SOM

L'algorithme Growing Cellular Probabilistic Self Organizing Map (GCPSOM, [140]) est un algorithme basé sur SOM permettant de quantifier un flux de données en temps réel. Par rapport aux approches antérieures [54, 24, 153, 4, 100, 29], les avantages de cet algorithme sont multiples :

- La taille de la carte est mise à jour en cours d'apprentissage, et n'a pas besoin d'être spécifiée à priori.
- L'apprentissage peut se poursuivre perpétuellement en s'adaptant aux changements de structure du flux de données.
- L'apprentissage est efficace, puisque le nombre de neurones est directement adapté aux données.

Cet algorithme est basé à la fois sur Growing Self Organizing Map (GSOM, [4]), capable d'adapter le nombre de neurones en cours d'apprentissage, et Cellular Probabilistic Self Organizing Map (CPSOM, [29]), une version probabiliste de SOM qui repose sur des bases mathématiques plus rigoureuses que SOM et qui est adaptée à l'apprentissage incrémental de données évolutives.

GCPSOM associe à chaque neurone i un taux d'apprentissage inverse (B_i) et une erreur cumulée (E_i). Lorsque cette erreur dépasse un seuil prédéfini (GT), elle est propagée aux neurones voisins si i n'est pas sur un bord de la carte, ou conduit à la création de nouveaux neurones dans le cas contraire. Les nouveaux neurones sont initialisés avec les mêmes prototypes w et le même taux d'apprentissage que le neurone i.

Le taux d'apprentissage inverse B_i associé à chaque neurone et d'autant plus grand que le neurone est un bon représentant des données présentées. Ainsi la mise à jour du prototype d'un neurone sera d'autant plus importante que le neurone est peu représentatif des données. Pour que B_i reste adapté à l'évolution du flux, un paramètre d'oubli

(γ, qui diminue au cours de l'apprentissage) réduit régulièrement les taux d'apprentissage inverses afin de réduire l'importance des données les plus anciennes.

Pour pouvoir s'adapter aux variations de la structure du flux, GCPSOM se caractérise par des périodes régulières d'"oubli" qui accroissent sa flexibilité toute les λ_3 itérations. Ainsi, toutes les λ_3 itérations, le facteur d'oubli γ est réinitialisé et l'erreur cumulée de tous les neurones est réduite pour permettre à la carte de se stabiliser avant de créer de nouveaux neurones. Puisque en cas d'apprentissage continu le nombre de neurones peut s'accroitre indéfiniment et qu'il n'y a pas de réduction du nombre de neurones au cours de l'apprentissage, la diminution de l'erreur cumulée toutes les λ_3 itérations est d'autant plus forte que le nombre de neurones N est proche d'un nombre limite χ fixé à priori.

1) **Initialisation** :
   - Initialiser le paramètre d'appariement et le facteur d'oubli :

$$\begin{aligned} \beta &= \beta^{initial} \\ \gamma &= \gamma^{initial} \end{aligned}$$

   - Initialiser le taux d'apprentissage, l'erreur cumulative et l'erreur maximale :

$$\begin{aligned} B_i &= 0 \ \forall i \in N \\ E_i &= 0 \ \forall i \in N \\ H &= 0 \end{aligned}$$

   - Initialiser la carte initiale (2x2 est suffisant) avec des prototypes choisis aléatoirement.
   - Initialiser la fonction de voisinage K_{ij} tel que $\sum_{j=1}^{N} K_{ij} = 1$.
   - Calculer le seuil de croissance GT en fonction du facteur de propagation SF choisi par l'utilisateur et de la dimension des données (d) :

$$GT = -d \times ln(SF)$$

2) **Sélectionner** une donnée x de la base de donnée (aléatoirement ou séquentiellement selon le type de donnée)
3) **Étape d'affectation**. Calculer les probabilités d'affectation de chaque neurone :

$$P_i\left(x\left(t\right)\right) = \frac{e^{\left(-\frac{\beta(t)}{2}\sum_{j=1}^{N} K_{ij}\|x(t)-w_j\|^2\right)}}{\sum_{n=1}^{N} e^{\left(-\frac{\beta(t)}{2}\sum_{j=1}^{N} K_{ij}\|x(t)-w_j\|^2\right)}}$$

4) **Étape d'adaptation**. Ajuster les prototypes de la carte :

$$\begin{aligned} w_i(t) &= w_i(t-1) + \frac{1}{B_i(t)} \sum_{j=1}^{N} K_{ij} P_j\left(x\left(t\right)\right) \left[x(t) - w_i(t-1)\right] \\ B_i(t) &= B_i(t-1) + \sum_{j=1}^{N} K_{ij} P_j\left(x\left(t\right)\right) \end{aligned}$$

5) **Étape d'accroissement** :

5.1) Soit u^* le neurone ayant la probabilité d'affectation la plus élevée pour $x(t)$, calculer l'erreur cumulative de u^* :

$$E_{u^*} = E_{u^*} + \frac{1}{2}\|x(t) - w_{u^*}\|^2$$

5.2) Si $E_{u^*} > H$ alors $H = E_{u^*}$.

5.3) Si $H > GT$ et que u^* est sur un bord de la carte, ajouter des neurones dans le voisinage de u^* tels que u^* ait quatre voisins immédiats, les prototypes et taux d'apprentissage sont initialisés aux valeurs de u^*.

5.4) Le cas échéant, normaliser K_{ij} tel que $\sum_{j=1}^{N} K_{ij} = 1$.

5.5) Si $H > GT$ et que u^* n'est pas sur un bord de la carte, propager l'erreur de u^* à ses voisins, en fonction du facteur de distribution FD choisi par l'utilisateur :

$$\begin{aligned} E_{u^*}(t+1) &= \frac{GT}{2} \\ E_i(t+1) &= E_i(t) + FD \times E_i(t), \forall i \text{ voisin de } u^* \end{aligned}$$

6) **Étape d'ajustement** des paramètres :

– Si $\beta(t) < \beta^{final}$ et $t \; mod \; \lambda_1 = 0$:

$$\beta(t+1) = \beta(t) + \Delta_1$$

– Si $\gamma(t) > 1$ et $t \; mod \; \lambda_2 = 0$:

$$\begin{aligned} \gamma(t+1) &= \gamma(t) - \Delta_2 \\ B_i(t+1) &= B_i(t)/\gamma(t+1) \; \forall i \in N \end{aligned}$$

– Si $t \; mod \; \lambda_3 = 0$:

$$\begin{aligned} \gamma(t+1) &= \gamma^{initial} \\ \beta(t+1) &= \beta^{initial} \\ H(t+1) &= H(t)\left(1 - \frac{N}{\chi}\right) \\ E_i(t+1) &= E_i(t)\left(1 - \frac{N}{\chi}\right) \; \forall i \in N \end{aligned}$$

7) Répéter les étapes 2 à 6.

5.4.2.2 Une adaptation pour le clustering de flux de données

Nous proposons dans cette section une adaptation de GCPSOM permettant d'effectuer un clustering des données du flux. Pour cela, les neurones de GCPSOM sont enrichis avec des informations supplémentaires apprises à partir des données, selon le principe de DS2L-SOM. Les informations associées à chaque neurone sont :

- L'effectif N_i. C'est le nombre de données ayant le neurone i pour meilleur représentant.
- La densité D_i. C'est une estimation de la densité des données dans le voisinage du prototype w_i.
- La variabilité cumulée s_i. C'est la somme des distances entre w_i et les données représentées par i.

De plus on associe une valeurs des connexions v_{ij} entre chaque paire de neurones i et j. Cette valeur représente le nombre de données correctement représentées à la fois par i et j. Toutes ces valeurs sont initialisées à zéro.

La mise à jour de ces valeurs se fait à chaque présentation d'une nouvelle donnée. Elles sont régulièrement pondérées par le facteur d'oubli γ, de façon à privilégier les infirmations nouvelles par rapport au plus anciennes et ainsi décrire l'état actuel du flux.

Lors de la création de nouveaux neurones dans le voisinage d'un neurone i, les prototypes de ces nouveaux neurones sont similaires au prototype w_i. De ce fait, les densités des nouveaux neurones sont égaux à D_i. De plus, les données initialement représentées par i sont maintenant aussi bien représentées par les nouveaux neurones. Les valeurs N_i et s_i sont alors réparties équitablement entre i et les nouveaux neurones (cf. l'algorithme). Pour finir, les valeurs des connexions entre i et ses anciens voisins sont réparties entre tous les voisins de i de façon à ce que la somme reste constante.

L'algorithme est le suivant :

1) **Initialisation** :
   - Initialiser le paramètre d'appariement et le facteur d'oubli :

$$\begin{aligned} \beta &= \beta^{initial} \\ \gamma &= \gamma^{initial} \end{aligned}$$

   - Initialiser le taux d'apprentissage, l'erreur cumulative et l'erreur maximale :

$$\begin{aligned} B_i &= 0 \ \forall i \in N \\ E_i &= 0 \ \forall i \in N \\ H &= 0 \end{aligned}$$

- Initialiser la carte initiale (2x2 est suffisant) avec des prototypes choisis aléatoirement.
- Initialiser la fonction de voisinage K_{ij} tel que $\sum_{j=1}^{N} K_{ij} = 1$.
- Initialiser l'effectif N_i, la densité D_i et la variabilité cumulée s_i de chaque neurone i à zéro.
- Initialiser les valeurs des connexions v_{ij} entre chaque paire de neurones i et j à zéro.
- Calculer le seuil de croissance GT en fonction du facteur de propagation SF choisi par l'utilisateur et de la dimension des données (d) :

$$GT = -d \times ln(SF)$$

2) **Sélectionner** une donnée x de la base de données (aléatoirement ou séquentiellement selon le type de données)

3) **Étape d'affectation**. Calculer les probabilités d'affectation de chaque neurone :

$$P_i\left(x\left(t\right)\right) = \frac{e^{\left(-\frac{\beta(t)}{2}\sum_{j=1}^{N} K_{ij}\|x(t)-w_j\|^2\right)}}{\sum_{n=1}^{N} e^{\left(-\frac{\beta(t)}{2}\sum_{j=1}^{N} K_{ij}\|x(t)-w_j\|^2\right)}}$$

4) **Étape d'adaptation**. Soit u^* et u^** les deux neurones ayant les probabilités d'affectation les plus élevées pour $x(t)$, adapter les paramètres suivants :

$$\begin{aligned}
w_i(t) &= w_i(t-1) + \frac{1}{B_i(t)}\sum_{j=1}^{N} K_{ij} P_j\left(x\left(t\right)\right)\left[x(t) - w_i(t-1)\right] \\
B_i(t) &= B_i(t-1) + \sum_{j=1}^{N} K_{ij} P_j\left(x\left(t\right)\right) \\
D_i(t) &= D_i(t-1) + e^{-\frac{\|x(t)-w_i(t-1)\|^2}{2\sigma^2}} \\
N_{u^*}(t) &= N_i(t-1) + 1 \\
s_{u^*}(t) &= s_{u^*}(t-1) + \|x(t) - w_i(t-1)\| \\
v_{u^*,u^{**}}(t) &= v_{u^*,u^{**}}(t-1) + 1
\end{aligned}$$

5) **Étape d'accroissement** :

5.1) Calculer l'erreur cumulative de u^* :

$$E_{u^*} = E_{u^*} + \frac{1}{2}\|x(t) - w_{u^*}\|^2$$

5.2) Si $E_{u^*} > H$ alors $H = E_{u^*}$.

5.3) Si $H > GT$ et que u^* est sur un bord de la carte, ajouter NV neurones dans le voisinage disponible de u^*. Pour tout neurone n ainsi crée :
- Initialiser les prototypes w_n, les densités D_n et les taux d'apprentissage B_n aux valeurs de u^*.
- Répartir les effectifs et normaliser les variabilités cumulées de u^* et des nouveaux neurones n : $N_i = N_{u^*}/(NV+1)$ et $s_i = s_{u^*}/(NV+1)$.
- Initialiser les valeurs de voisinages des nouveaux neurones :

$$v_{n,u^*} = N_{u^*}/NV$$

.
- Normaliser les valeurs des connexions entre u^* et ses voisins anciens (v) et nouveaux (n) :

$$v_{i,u^*} = v_{i,u^*} \frac{\sum v_{v,u^*}}{\sum v_{v,u^*} + \sum v_{n,u^*}} \ \forall i \in \{v, n\}$$

5.4) Le cas échéant, normaliser K_{ij} tel que $\sum_{j=1}^{N} K_{ij} = 1$.

5.5) Si $H > GT$ et que u^* n'est pas sur un bord de la carte, propager l'erreur de u^* à ses voisins, en fonction du facteur de distribution FD choisi par l'utilisateur :

$$\begin{aligned} E_{u^*}(t+1) &= \frac{GT}{2} \\ E_i(t+1) &= E_i(t) + FD \times E_i(t), \forall i \text{ voisin de } u^* \end{aligned}$$

6) **Étape d'ajustement** des paramètres :
- Si $\beta(t) < \beta^{final}$ et $t \ mod \ \lambda_1 = 0$:

$$\beta(t+1) = \beta(t) + \Delta_1$$

- Si $\gamma(t) > 1$ et $t \ mod \ \lambda_2 = 0$:

$$\begin{aligned} \gamma(t+1) &= \gamma(t) - \Delta_2 \\ B_i(t+1) &= B_i(t)/\gamma(t+1) \ \forall i \in N \\ N_i(t+1) &= N_i(t)/\gamma(t+1) \ \forall i \in N \\ D_i(t+1) &= D_i(t)/\gamma(t+1) \ \forall i \in N \\ s_i(t+1) &= s_i(t)/\gamma(t+1) \ \forall i \in N \\ v_{ij}(t+1) &= v_{ij}(t)/\gamma(t+1) \ \forall i,j \in N^2 \end{aligned}$$

– Si $t \ mod \ \lambda_3 = 0$:

$$\begin{aligned} \gamma(t+1) &= \gamma^{initial} \\ \beta(t+1) &= \beta^{initial} \\ H(t+1) &= H(t)\left(1-\frac{N}{\chi}\right) \\ E_i(t+1) &= E_i(t)\left(1-\frac{N}{\chi}\right) \ \forall i \in N \end{aligned}$$

7) Répéter les étapes 2 à 6.

Toutes les λ_3 itérations, la carte SOM enrichie représente l'état actuel du flux. Elle peut être enregistrée dans un espace de stockage est peut être traitée selon les besoins par un algorithme de clustering (voir l'algorithme de raffinement de la section 2.3.1.2) ou par l'algorithme d'estimation de la densité sous-jacente proposé au chapitre 4.2, etc ...

La comparaison de deux enregistrements effectués à des périodes différentes, par exemple avec la méthode de la section 4.4.3, permettent d'analyser les variations de la structure du flux entre ces deux périodes. Ainsi, il devient possible de comprendre la dynamique du flux à différentes échelles temporelles.

5.4.3 Compression et stockage de la structure du flux

Une difficulté de l'analyse d'un flux de données est que ce flux est potentiellement infini. On ne peut donc pas se permettre de stoker indéfiniment des cartes SOM représentant différents instants, puisque les capacités de stockage sont limitées. Nous souhaitons donc proposer une méthode de fusion de SOM enrichies, qui permettra de compresser l'information stockée.

L'idée est de fusionner deux SOM ou plus représentant des instants successifs si la structure des données représentées par ces deux cartes est suffisamment similaire. Nous proposons, si la puissance de calcul disponible le permet, de construire et de mettre à jours une matrice de similarité entre les fonctions de densité représentant la structures du flux pour des périodes consécutives. Pour cela il suffit de comparer chaque nouvelle SOM enrichie stockée avec la SOM la plus récente déjà stockée. Puis, si la place manque, il est possible de fusionner les deux SOM adjacentes les plus similaires. Ainsi, à chaque compression des informations stockées, on conserve le maximum d'information sur les variations du flux.

La fusion de SOM peut se faire en générant des données à partir des fonctions de densités et en lançant l'algorithme SOM enrichi sur ces données (cf. section 4.2.1).

Soit N SOM enrichies et leur fonction de densité : $SOM^1 = \{N_i^1, w_i^1, \alpha_i^1, h_i^1\}_{i=1}^{M^1}$, ..., $SOM^N = \{N_i^N, w_i^N, \alpha_i^N, h_i^N\}_{i=1}^{M^N}$. L'algorithme de génération des données est le suivant :

1) Sélectionner aléatoirement une des SOM. Chaque SOM A, composée de M^A neurones, a une probabilité d'être choisie de :

$$P(A) = \frac{\sum_{i=1}^{M^A} N_i^A}{\sum_{K=1}^{N} \sum_{i=1}^{M^K} N_i^K}$$

Autrement dit, plus une SOM représente une grande quantité de données, plus elle a de chance d'être sélectionnée.

2) Sélectionner aléatoirement un neurone i de la SOM choisie en fonction du paramètre α, qui représente la contribution du neurone vis-à-vis de la fonction de densité (cf. section 4.2.2) :

$$P(i) = \frac{\alpha_i}{\sum_{j=1}^{M} \alpha_j}$$

3) Générer une donnée aléatoirement selon une distribution gaussienne sphérique centrée sur w_i et d'écart-type h_i.

Il suffit alors d'appliquer un algorithme SOM enrichi sur les données générées et d'estimer une fonction de densité pour obtenir une représentation condensée de la structure des SOM d'origine.

Dans cette partie nous avons proposé une méthode générale d'analyse de la dynamique de la structure d'un flux de données. Cette méthode s'appuie largement sur les travaux présentés dans les parties précédentes. Nous avons introduit en outre un nouvel algorithme de suivi de la structure d'un flux, ainsi qu'une méthode de compression des informations stockées.

5.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés aux données dynamiques. Nous avons présenté deux exemples d'applications pour la trajectométrie. Les résultats obtenus

sont cohérents avec les connaissances des experts du domaine et la méthode proposée semble pertinente.

Nous avons aussi proposé une méthode d'analyse de grands flux de données. Bien que reposant sur des algorithmes ayant déjà été testés, cette méthode n'a pas encore été validée par des expériences sur des bases de données artificielles ou réelles. Une telle validation est nécessaire et devrait être effectuée dans le futur pour vérifier la qualité de la méthode proposée.

Conclusions et perspectives

Dans cette thèse nous avons présenté plusieurs nouveaux algorithmes d'analyse et de modélisation de la structure d'un ensemble de données. Tous ces algorithmes sont dérivés des algorithmes de classification à deux niveaux et sont basés sur l'apprentissage d'une carte auto-organisatrice.

Pour la classification non supervisée de données vectorielles, nous avons proposé des algorithmes qui ont de nombreux avantages par rapport à leurs principaux concurrents pour la découverte de clusters naturels dans les données. L'utilisation d'une étape de quantification des données par une SOM, combinée à l'apprentissage, en même temps que la mise à jour des prototypes, d'informations sur la connectivité et de la densité, permet la détection de clusters de formes arbitraires, bruités, en contact et/ou non linéairement séparables. De plus, la vitesse d'exécution est linéaire selon le nombre de données. Les paramètres des algorithmes n'ont pas besoin d'être choisis séparément pour différents jeux de données. En particulier, le nombre de clusters à obtenir est détecté automatiquement par ces algorithmes en fonction des informations apprises à partir des données.

Pour les données non-vectorielles, nous avons développé une extension des algorithmes précédents pour des données de type intervalles. Les bons résultats obtenus montrent qu'il est possible d'adapter ces algorithmes pour l'analyse de données non-vectorielles en modifiant la définition d'un prototype et la similarité entre une donnée et un prototype.

Par ailleurs, nous avons montré qu'il est possible d'améliorer significativement la qualité de la quantification des données par une SOM en tenant en compte des informations de connectivité entre prototypes pour optimiser leur mise à jour. Les résultats obtenus montrent que le compromis quantification-visualisation est amélioré par cette méthode.

Les algorithmes proposés pour estimer et comparer les distributions des données présentent des caractéristiques intéressantes pour l'analyse de grandes bases de données ou de flux de données. Il s'agit aussi de méthodes à deux niveaux, ce qui permet une analyse en ligne des données, rapide et sans surcharge de mémoire. Les résultats obtenus sont de plus de bonne qualité.

Nous avons en outre proposé une nouvelle façon de mesurer la stabilité de nos algorithmes, en s'appuyant sur la comparaison des densités des différents clusters. Les performances de cette mesure pour la sélection des dimensions de la carte ont été

comparées avec les performances d'une mesure classique de stabilité et d'une sélection heuristique de ces paramètres. Nous avons montré que les trois méthodes se valent pour la sélection de paramètres et que DS2L-SOM est capable de bonnes performances pour une gamme relativement large de dimensions de la carte, malgré une sensibilité non négligeable au ratio longueur/largeur.

De plus, nous avons proposé une méthode générale d'analyse de la dynamique de la structure d'un flux de données. Nous avons introduit en outre un nouvel algorithme de suivi de la structure d'un flux, ainsi qu'une méthode de compression des informations stockées.

Enfin, deux applications réelles ont été présentées dans cette thèse. Toutes deux reposent sur l'analyse de trajectoires spatio-temporelles issues de la technologie RFID. Cette technologie récente connait un fort développement et il existe à ce jour peu de méthodes pour analyser ce type d'enregistrements très volumineux et fortement bruités. Les résultats obtenus avec nos algorithmes sont cohérents avec les connaissances des experts du domaine, ce qui conforte l'utilisation de ce type de méthode.

Plusieurs perspectives de recherche peuvent être envisagées suite à ces travaux :

1) **Analyse de données non vectorielles selon la densité**. La découverte de clusters selon la densité n'a pour l'instant été testé que pour des données vectorielles. Pourtant, il est tout à fait possible de définir une densité pour d'autres type de données, et d'utiliser cette information pour obtenir une partition de ces données.
2) **Version à noyaux et matrice de similarité**. Il est très fréquent que les données ne soient définies que par leur similarités réciproques. Souvent, lorsque les données sont complexes (images, texte, ...), un noyau est proposé comme mesure de similarité entre ces objets. Il existe quelques méthodes de classification basées sur un noyau ou une matrice de similarité. En particulier, des extensions de SOM ont été proposées dans ce sens. Il serait ainsi intéressant d'étendre les algorithmes de classification proposés dans cette thèse pour l'analyse de matrice de similarité.
3) **Étude plus approfondie de l'estimation de la distribution**. La méthode à deux niveaux est rapide et semble performante. Pourtant, elle repose sur un certain nombre d'heuristiques qui pourrait faire l'objet d'une analyse et d'une formalisation plus poussée. De plus, une comparaison directe avec des méthodes basées sur l'algorithme EM pourrait souligner les avantages de notre algorithme (en particulier en terme de rapidité) par rapport à ses concurrents.
4) **Validation de l'algorithme d'analyse des flux de données**. Nous avons proposé une extension des algorithmes de classification aux flux de données. Ce-

pendant, cette proposition n'a pas été testée et aucune expérience sur la qualité des résultats ne nous permet de valider l'approche. De tels expériences sont nécessaires et doivent être effectuées dans le futur. De plus, la méthode est basée sur un algorithme de quantification pour flux de données qui présente de nombreux défauts (en particulier un nombre important de paramètres à sélectionner). Nous souhaitons proposer une nouvelle extension de SOM adaptée à l'analyse des flux de données comme base plus solide à une classification des données du flux.

5) **Sélection de modèles**. Enfin, nos travaux sur la sélection de modèles en sont encore à leur début. Nous envisageons de travailler à la validation du principe de stabilité pour le choix de la taille de la carte à utiliser dans les approches basées sur l'apprentissage d'une SOM.

Annexe A

Autres travaux réalisés

A.1 Bi-clustering

Il est parfois très intéressant de pouvoir regrouper et visualiser les attributs servant à décrire les données en plus de la classification de ces données. Cela permet, par exemple, d'associer de façon simple chaque regroupement de données avec les attributs caractéristiques de ce regroupement, mais aussi de visualiser des corrélations entre les attributs.

A.1.1 Kdisj

L'algorithme Kdisj [34] est une adaptation de SOM qui permet de projeter sur la carte à la fois les données et les attributs servant à les décrire (c'est l'équivalent non linéaire de l'ACM [93, 75]). Il est conçu pour la classification de données qualitatives, sous la forme d'un tableau disjonctif D. Un attribut présente plusieurs modalités mutuellement exclusives (par exemple l'attribut « couleur » peut présenter les modalités « jaune », « vert », etc...) qui peuvent être codées sous la forme d'un vecteur de dimensions égales au nombre de modalités et de valeurs nulles dans toutes les dimensions sauf une. On peut coder de la même manière plusieurs attributs par un vecteur de taille égale à l'ensemble des modalités des différents attributs avec autant de valeurs non nulles que d'attributs (cf Table A.1 par exemple).

L'idée est que l'on peut décrire une donnée en fonction des modalités associées (vecteur ligne), mais que l'on peut aussi décrire une modalité en fonction des données

TABLE A.1 – Exemple de tableau disjonctif pour données qualitatives

	Couleur				Taille		
Données	Jaune	Vert	Bleu	Rouge	Petit	Moyen	Grand
1	0	1	0	0	0	1	0
2	0	0	0	1	0	0	1
3	1	0	0	0	0	1	0
4	0	0	1	0	1	0	0

associées (vecteur colonne). L'ensemble données + modalités peut alors être représenté dans un espace de dimension A + E (nombre de modalités des Attributs + nombre de données ou Exemples d'apprentissage) et on peut faire évoluer une carte auto-organisatrice dans cette espace en lui présentant au cours de l'apprentissage alternativement une donnée et une modalité. La distance entre une donnée (taille A) et un prototype de la carte (taille A + E) sera calculée sur les A premières dimensions, alors que la distance entre une modalité (taille E) et un prototype sera calculée sur les E dernières. De façon à assurer un lien entre les A premières dimensions et les E dernières, le prototype sera adapté sur toutes ses dimensions lors de la phase d'adaptation, en associant à la donnée présentée sa modalité non nulle la plus caractéristique (c'est à dire la plus rare dans l'ensemble des données). On adapte ainsi les A premières dimensions du prototype en fonction de la donnée présentée et les E dernières en fonction de la modalité associée. Il faut noter qu'il n'est pas possible de faire de même lorsqu'on présente une modalité, puisqu'il n'y a pas de données rares dans la description de l'ensemble des modalités (chaque donnée est caractéristique d'exactement autant de modalités qu'il y a d'attributs).

Algorithme KDisj :

1) **Initialisation** :
   - Définir la topologie de la carte.
   - Préparer les données sous forme de tableaux disjonctifs : $D = A \times E$, avec A le nombre de modalités des attributs et E le nombre d'exemples.
   - Corriger le tableau disjonctif D en D_c :

$$d_{ij}^c = \frac{d_{ij}}{\sqrt{d_{i.}d_{.j}}} \text{ avec } d_{i.} = \sum_{j=1}^{N} d_{ij} \text{ et } d_{.j} = \sum_{i=1}^{N} d_{ij}$$

     De cette façon, utiliser la distance euclidienne sur le tableau D_c (tableau disjonctif corrigé) équivaut à utiliser la distance du χ^2 pondérée sur D.
   - Initialiser aléatoirement tous les prototypes $w_j = (w_{Aj}, w_{Ej})$.

2) **Première phase de compétition** :
   - Présenter un exemple $x^{(k)}$ (soit une ligne de D_c) choisi aléatoirement.
   - Associer à $x^{(k)}$ la modalité $y(x^{(k)})$ définie par $y(x^{(k)}) = Argmax_y\, d^c_{xy}$ et créer le vecteur $X = (x^{(k)}, y(x^{(k)})) = (X_A, X_E)$.
   - Parmi les M neurones, choisir le meilleur représentant $u^*(x^{(k)})$ sur les A premières composantes :

$$u^*(x^{(k)}) = \underset{1 \leq i \leq M}{Argmin} \parallel X_A - w_{Ai} \parallel^2$$

3) **Première phase d'adaptation** :
   - Mettre à jour les prototypes w_j de chaque neurone j sur l'ensemble des composantes :

$$w_j(t) = w_j(t-1) - \varepsilon(t) K_{ju^*(x^{(k)})}(w_j(t-1) - X)$$

4) **Deuxième phase de compétition** :
   - Présenter un exemple $y^{(k)}$ (soit une colonne de D_c) choisi aléatoirement.
   - Créer le vecteur $Y = (y^{(k)}) = (Y_E)$.
   - Parmi les M neurones, choisir le meilleur représentant $u^*(y^{(k)})$ sur les E dernières composantes :

$$u^*(y^{(k)}) = \underset{1 \leq i \leq M}{Argmin} \parallel Y_E - w_{Ei} \parallel^2$$

5) **Deuxième phase d'adaptation** :
   - Mettre à jour les prototypes w_j de chaque neurone j sur les E dernières composantes :

$$w_{Ej}(t) = w_{Ej}(t-1) - \varepsilon(t) K_{ju^*(y^{(k)})}(w_{Ej}(t-1) - Y_E)$$

6) **Répéter les phases 2 à 5** jusqu'à ce que les mises à jours des prototypes soient négligeables.

A.1.2 S2L-KDisj : Une extension de KDisj

A.1.2.1 Algorithme

L'adaptation de KDisj pour la classification à deux niveaux simultanés est similaire à l'adaptation de SOM. A chaque fois que l'on présente une donnée ou une modalité on augmente la valeur de la connexion entre les deux prototypes les plus sensibles et on diminue celle des autres connexions. La version présentée ici est, de plus, adaptée à

des données exprimées en fréquence ou en proportion, c'est à dire que l'on associe un pourcentage à chaque modalité d'un attribut (voir Table A.2), la somme des modalités pour un attribut étant égale à 1 (ou 100%). Ce type de données est très utilisé dans de nombreux domaines (gestion du temps, budget, modalités variables dans le temps ou l'espace,...). La seule différence, dans ce cas, avec un tableau disjonctif est que l'on peut associer une donnée caractéristique à une modalité, lors des étapes 4 et 5.

TABLE A.2 – Exemple de tableau en fréquence

Données	Couleur (% de surface couverte)				Taille (% de temps passé dans cette position)		
	Jaune	Vert	Bleu	Rouge	Horizontal	Vertical	Diagonal
1	10	80	7	3	20	50	30
2	4	1	1	94	52	38	10
3	23	61	2	14	12	43	45
4	5	25	67	3	20	13	67

Algorithme S2L-KDisj

1) **Initialisation** :
   - Définir la topologie de la carte.
   - Préparer les données sous forme de tableaux disjonctifs : $D = A \times E$, avec A le nombre de modalités des attributs et E le nombre d'exemples.
   - Corriger le tableau disjonctif D en D_c :

$$d^c_{ij} = \frac{d_{ij}}{\sqrt{d_{i.}d_{.j}}} \text{ avec } d_{i.} = \sum_{j=1}^{N} d_{ij} \text{ et } d_{.j} = \sum_{i=1}^{N} d_{ij}$$

     De cette façon, utiliser la distance euclidienne sur le tableau D_c (tableau disjonctif corrigé) équivaut à utiliser la distance du χ^2 pondéré sur D.
   - Initialiser aléatoirement tous les prototypes $w_j = (w_{Aj}, w_{Ej})$.
   - Initialiser les connexions ν entre chaque couple de neurones i et j :

$$\forall i, j \quad \nu_{ij} = 0$$

2) **Première phase de compétition** :
   - Présenter un exemple $x^{(k)}$ (soit une ligne de D_c) choisi aléatoirement.
   - Associer à $x^{(k)}$ la modalité $y(x^{(k)})$ définie par $y(x^{(k)}) = Argmax_y\, d^c_{xy}$ et créer le vecteur $X = (x^{(k)}, y(x^{(k)})) = (X_A, X_E)$.

– Parmi les M neurones, choisir les deux meilleurs représentants $u^*(x^{(k)})$ et $u^*(x^{(k)})$ sur les A premières composantes :

$$u^*(x^{(k)}) = \underset{1\leq i\leq M}{Argmin} \parallel X_A - w_{Ai} \parallel^2$$
$$u^{**}(x^{(k)}) = \underset{1\leq i\leq M, i\neq u^*}{Argmin} \parallel X_A - w_{Ai} \parallel^2$$

– Mettre à jours les valeurs de connexion entre $u^*(x^{(k)})$ et ses voisins :

$$\nu_{u^*u^{**}}(t) = \nu_{u^*u^{**}}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*u^{**}}(t-1) - 1\right)$$
$$\nu_{u^*i}(t) = \nu_{u^*i}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*i}(t-1)\right) \quad \forall i \text{ voisin de } u^*$$

Avec :

$$r(t) = \frac{1}{1 + e^{-\left(\frac{t}{t_{max}}\right)}}$$

3) **Première phase d'adaptation** :

– Mettre à jour les prototypes w_j de chaque neurone j sur l'ensemble des composantes :

$$w_j(t) = w_j(t-1) - \varepsilon(t)K_{ju^*(x^{(k)})}(w_j(t-1) - X)$$

4) **Deuxième phase de compétition** :

– Présenter un exemple $y^{(k)}$ (soit une colonne de D_c) choisi aléatoirement.

– Associer à $y^{(k)}$ la modalité $x(y^{(k)})$ défini par $x(y^{(k)}) = Argmax_x\, d^c_{xy}$ et créer le vecteur $Y = (x(y^{(k)}), y^{(k)}) = (Y_A, Y_E)$.

– Parmi les M neurones, choisir les deux meilleurs représentants $u^*(y^{(k)})$ et $u^{**}(y^{(k)})$ sur les E dernières composantes :

$$u^*(y^{(k)}) = \underset{1\leq i\leq M}{Argmin} \parallel Y_E - w_{Ei} \parallel^2$$
$$u^{**}(y^{(k)}) = \underset{1\leq i\leq M, i\neq u^*}{Argmin} \parallel Y_E - w_{Ei} \parallel^2$$

– Mettre à jours les valeurs de connexion entre $u^*(y^{(k)})$ et ses voisins :

$$\nu_{u^*u^{**}}(t) = \nu_{u^*u^{**}}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*u^{**}}(t-1) - 1\right)$$
$$\nu_{u^*i}(t) = \nu_{u^*i}(t-1) - \varepsilon(t)r(t)\left(\nu_{u^*i}(t-1)\right) \quad \forall i \text{ voisin de } u^*$$

Avec :

$$r(t) = \frac{1}{1 + e^{-\left(\frac{t}{t_{max}}\right)}}$$

5) **Deuxième phase d'adaptation** :
   - Mettre à jour les prototypes w_j de chaque neurone j sur l'ensemble des composantes :
   $$w_j(t) = w_j(t-1) - \varepsilon(t) K_{ju^*(y^{(k)})}(w_j(t-1) - Y)$$
6) **Répéter les phases 2 à 5** jusqu'à ce que les mises à jour des prototypes soient négligeables.

A.1.2.2 Validation

De façon à évaluer les performances de S2L-KDisj, nous avons utilisé les données et les résultats d'une étude portant sur la gestion de l'espace et du temps par les individus d'une fourmilière en laboratoire [50]. Une reine (R), un mâle (M), un jeune (J) et 43 ouvrières (2 à 44) ont été observés dans une fourmilière composée de 9 salles (Loc2 à Loc10), un couloir menant à l'extérieur (Loc1) et une zone de récolte (Loc0, cf Figure A.1).

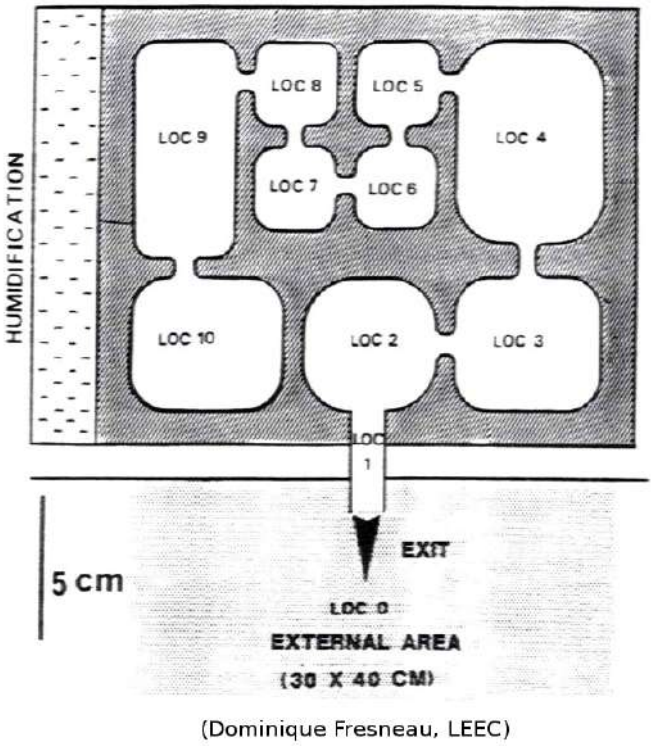


FIGURE A.1 – Disposition des salles dans la fourmilière.

Pour chaque individu, nous connaissons la proportion de temps passé dans chaque salle et dans différentes activités, grâce à une série de photos de l'ensemble des individus dans la fourmilière. Le traitement de ces données par des méthodes classiques (ACM et étude comportementale à partir de tests statistiques, [50]) sont représentés sur la figure A.2.

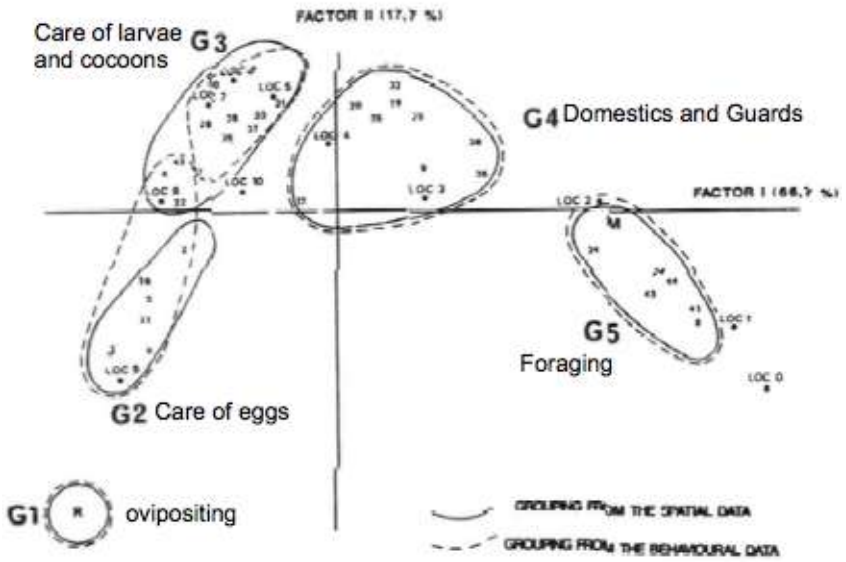


FIGURE A.2 – Résultats de l'étude par ACM et tests statistiques.

Les résultats obtenus par S2L-KDisj à partir des mêmes données sont représentés par la figure A.3. Les salles Loc0 à Loc10 sont notées C0 à C10. Les comportements sont notés par un code de trois lettres, la dernière donnant la catégorie générale (T : entrée et sortie de la fourmilière, N : gestion de la nourriture, C : soins aux cocons, L : soins aux larves, O : soins aux œufs).

Les résultats sont très similaires. Les groupes G1 et G2 de l'étude classique regroupent la reine, le jeune et quelques autres individus dans la salle 9 avec des comportements de soins aux œufs. Le groupe « bleu » de S2L-KDisj regroupe les mêmes individus dans la salle 9 avec des comportements d'immobilité sur les œufs et les larves. Le groupe G3 regroupe les même individus que le groupe « vert » avec les salles 5, 6, 7, 8 et 10 et des comportements de soins aux larves et cocons. Le groupe « jaune » est identique au groupe G4 et le groupe « rouge » au groupe G5. Le groupe « orange » n'existe pas dans les résultats originaux, il regroupe le mâle dans les salles 0 et 1 (le couloir et l'extérieur) avec des comportements d'entrée et sortie de la fourmilière.

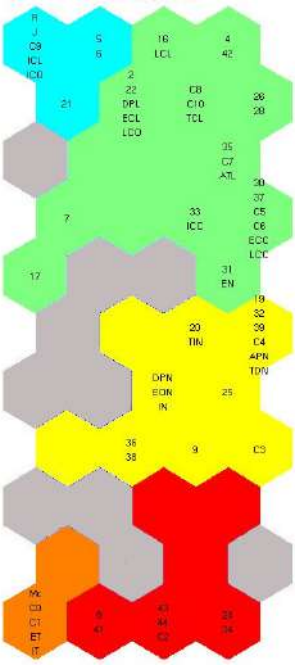


FIGURE A.3 – Résultats de l'étude par S2L-KDisj.

Ainsi, les résultats obtenus par S2L-KDisj sont similaires en quantité et en qualité à ceux obtenus à l'aide d'une double étude statistique qui a demandé une quantité importante de travail aux biologistes, mais en une seule étape et en quelques secondes.

A.2 Version Batch généralisée

Nous proposons dans cette section une version Batch généralisée de l'algorithme de clustering. Il procède par étapes successives au lieu de tout apprendre en même temps. On l'a vu, la fonction de coût que l'on cherche à minimiser dans DS2L-SOM est une combinaison de trois fonctions de coût :

– Le coût lié aux prototypes (ici pour SOM) :

$$\widetilde{R}(w) = \sum_{k=1}^{N} \sum_{j=1}^{M} K_{ju^*(x^{(k)})} \|w_j - x^{(k)}\|^2$$

– Le coût lié à l'estimation des densités :

$$\widetilde{R}(D) = \sum_{k=1}^{N} \sum_{j=1}^{M} \left[D_j - e^{\frac{\|w_j - x^{(k)}\|^2}{2\sigma^2}} \right]^2$$

– Le coût lié à l'estimation des valeurs de connexions :

$$\widetilde{R}(v) = \sum_{k=1}^{N} \sum_{i,j=1}^{M} \left[v_{ij} - \mathbb{1}_{\{w_i, w_j \text{ bmus de } x^{(k)}\}} \right]^2$$

On voit bien ici qu'une fois les prototypes w calculés de façon à minimiser $\widetilde{R}(w)$, il suffit de définir D et v (qui dépendent de w) telles que $\widetilde{R}(D)$ et $\widetilde{R}(v)$ soient minimales.

De ce fait, les solutions obtenues seront plus fiable que celles de l'algorithme stochastique (chapitre 2.3), qui ne minimise pas complètement $\widetilde{R}(D)$ et $\widetilde{R}(v)$. Cette algorithme est de plus très bien adapté à l'analyse de données complexes, pouvant être décrites par exemple par l'intermédiaire d'une matrice de dissimilarité ou d'un noyau. Cependant, cet algorithme ne fonctionne pas en ligne, et il est nécessaire de conserver en mémoire les similarités entre les données et les prototypes. Il n'est donc pas adapté à l'analyse de grandes bases de données ou de flux de données.

A.2.1 Apprentissage des prototypes

La première étape de l'algorithme est l'apprentissage des prototypes. La méthode utilisée n'est pas très importante, à condition que le nombre de prototypes soit suffisant. Le choix de l'algorithme dépend surtout de l'application. Les méthodes à base de SOM permettent une visualisation simple et efficace de la structure des données, mais il est aussi possible d'utiliser des algorithmes comme Neural Gas qui ne contraint pas la topologie de l'ensemble des prototypes. Il existe de plus de nombreuses variantes de ce type d'algorithmes, adaptées à tous types de données. En particulier les méthodes à base de noyaux ou de matrice de dissimilarité (pour des méthodes basées sur SOM voir [160, 148, 31, 21]) permettent d'analyser la structure de données complexes (images, texte, ...).

Algorithme général d'apprentissage des prototypes :

1) Les prototypes sont appris selon un algorithme au choix (par exemple la version Batch de SOM, Neural Gas ou une méthode à noyaux).
2) L'algorithme retourne $d(w, x)$, la matrice des dissimilarités entre chaque prototype et chaque donnée.

A.2.2 Enrichissement des prototypes

La deuxième étape est l'extraction des informations structurelles à partir des relations entre données et prototypes.

L'algorithme d'enrichissement procède en trois étapes :

Entrées :
- La matrice $d(w, x)$ des distances entre les M prototypes w et les N données x.

Sorties :
- Une estimation de la densité D_j associée à chaque prototype j.
- Une estimation des valeurs de voisinage $v_{i,j}$ associées à chaque paire de prototype i et j.

Algorithme :
- **Estimation de la densité** :

$$D_j = 1/N \sum_{x=1}^{N} e^{-\frac{d(w_j,x)^2}{2\sigma^2}}$$

  avec σ un paramètre à choisir par l'utilisateur.
- **Estimation des valeurs de voisinage** :
  - Pour chaque donnée x, trouver les deux prototypes les plus proches (BMUs : Best Match Units) $u^*(x)$ et $u^{**}(x)$:

$$u^*(x) = argmin_i(d(w_i, x)) \text{ et } u^{**}(x) = argmin_{i \neq u^*(x)}(d(w_i, x))$$

  - Calculer $v_{i,j}$ = le nombre de données ayant i et j comme deux premiers BMUs.

A la fin de cette étape, à chaque prototype est associée une valeur de densité et à chaque paire de prototypes est associée une valeur de voisinage. Une grande partie de l'information sur la structure des données est stockée dans ces valeurs. Il n'est plus nécessaire de garder les données en mémoire.

A.2.3 Clustering automatique des prototypes

La dernière étape est le clustering des prototypes. Cette étape utilise les informations calculées lors de l'étape précédente. L'algorithme est exactement le même que l'algorithme de raffinement présenté dans la section 2.3.1.2.

Algorithme de raffinement :

Entrées :
– Les valeurs de densité D_j et de voisinage $v_{i,j}$.
Sorties :
– Des groupes de prototypes similaires (les *clusters*).

1) **Extraire tous les ensembles de prototypes connectés** :
Soit $P = \{C_i\}_{i=1..L}$ les L ensembles de prototypes inter-connectées : $\forall m \in C_i, \exists n \in C_i$ tel que $v_{m,n} > seuil$.

2) **Pour chaque groupe** $C_k \in P$ **faire** :
   – Déterminer l'ensemble $M(C_k)$ des maximums locaux de densité (les modes de densité) :
   $$M(C_k) = \{i \in C_k \mid D_i \geq D_j, \forall j \text{ voisin de } i\}$$
   – Calculer la matrice des seuils :
   $$S = [S(i,j)]_{i,j=1...|M(C_k)|}$$
   avec
   $$S(i,j) = \left(\frac{1}{D_i} + \frac{1}{D_j}\right)^{-1}$$
   – Pour chaque $i \in C_k$, étiqueter i avec un élément $label(i)$ de $M(C_k)$, selon un gradient ascendant de densité le long des connexions de voisinage (reliant deux prototypes m et n si $v_{m,n} > seuil$). Chaque étiquette représente un sous-groupe.
   – Pour chaque paire de prototypes voisins (i,j) dans C_k, si $label(i) \neq label(j)$ et si $D_i > S(label(i), label(j))$ et $D_j > S(label(i), label(j))$ alors fusionner les deux sous-groupes.

3) **Retourner les groupes fusionnés** (les *clusters*).

Bien que les premiers résultats de l'algorithme Batch aient montré la qualité et la pertinence de cette approche, une validation expérimentale plus poussé doit être réalisée. En particulier il faut tester la validité de la méthode pour différents algorithmes d'apprentissage des prototypes (SOM Batch, SOM à noyaux, ...).

Publications effectuées pendant la thèse

Revues Internationales

Cabanes, G., and Bennani, Y. : Unsupervised topographic learning for spatio-temporal data mining. Advances in Artificial Intelligence. Sous presse (2010). *Sur invitation*.

Cabanes, G., Bennani, Y., and Fresneau, D. : Mining RFID Behavior Data using Unsupervised Learning. International Journal of Applied Logistics. 1 (1), pp. 28–47 (2010).

Galassi, U., Cabanes, G., and Fresneau, D. : Modelling Evolving Behaviours in Ant Colonies. Journal of Intelligent Systems. 18 (4), pp. 353–376 (2009).

Chapitre de Livre

Cabanes, G. and Bennani, Y. : Learning the number of clusters in Self Organizing Map. In Self-Organizing Maps, George K Matsopoulos (Ed.), IN-TECH Publisher, pp. 14–28 (2010). *Sur invitation*.

Conférences Internationales

Cabanes, G., Bennani, Y. : Learning topological constraints in Self-Organizing Map. 17th International Conference on Neural Information Processing (ICONIP'10), Sydney, Australia (2010).

Cabanes, G., Bennani, Y. : Extending SOM with Efficient Estimation of the Number of Clusters. Joint Meeting GfKl-CLADAG'10, pp. 233–234, Florence, Italy (2010). *Sur invitation*.

Cabanes G., Bennani Y. and Dufau-Joël, F. : Mining Customers' Spatio-temporal Behavior Data using Topographic Unsupervised Learning. Proceedings of the International Conference on Machine Learning and Applications (ICMLA'09), pp. 372–377, Miami Beach, Florida, USA (2009).

Cabanes, G., Bennani, Y. : Comparing Large Datasets Structures Through Unsupervised Learning. 16th International Conference on Neural Information Processing (ICONIP'09), pp. 546–553, Bangkok, Thailand (2009). *Prix du meilleur poster étudiant*.

Cabanes, G., Fresneau, D., Galassi, U., and Giordana, A. : A HMM-Based Approach to Modeling Ant Behavior. 4th International Indian Conference on Artificial Intelligence (IICAI'09), pp. 2127–2139, Tumkur, India (2009). *Prix du meilleur article*.

Cabanes, G., Fresneau, D., Giordana, A., and Galassi, U. : Modeling Ant Activity by means of Structured HMMs. International Symposium on Methodologies for Intelligent Systems (ISMIS'09), pp. 341–350, Prague, Czech Republic (2009).

Cabanes, G., Bennani, Y., Chartagnat, C. and Fresneau, D. : Topographic Connectionist Unsupervised Learning for RFID Behavior Data Mining. The Second International Workshop on RFID Technology (IWRT'08), pp. 63–72, Barcelona, Spain (2008).

Cabanes, G., Bennani, Y. : A Local Density-Based Simultaneous Two-Level Algorithm for Topographic Clustering. Proceeding of the International Joint Conference on Neural Networks (IJCNN'08), pp. 1176–1182, Hong Kong, China (2008).

Cabanes, G., Bennani, Y. : A simultaneous two-level clustering algorithm for automatic model selection. Proceeding of the International Conference on Machine Learning and Applications (ICMLA'07), pp. 316–321, Cincinnati, Ohio, USA (2007).

Conférences Nationales

Cabanes, G., Bennani, Y. and Dufau-Joël, F. : Comparaisons structurelles de grandes bases de données par apprentissage non-supervisé. 10èmes journées d'Extraction et Gestion des Connaissances (EGC'10), pp. 115–120, Hammamet, Tunisie (2010).

Cabanes, G., Bennani, Y. : Exploration de données de traçabilité issues de la RFID par apprentissage non-supervisé. 9èmes journées d'Extraction et Gestion des Connaissances (EGC'09), pp. 451–452, Strasbourg, France (2009).

Cabanes, G., Bennani, Y. : Classification topographique à deux niveaux simultanés à base de modes de densité. Actes de la Conférence francophone sur l'Apprentissage automatique (CAP'08), pp. 53–54, Porquerolle, France (2008).

Cabanes, G., Bennani, Y. : Un algorithme de classification topographique non supervisée à deux niveaux simultanés. 8èmes journées d'Extraction et Gestion des Connaissances (EGC'08), pp. 619–630, Sophia-Antipolis, France (2008). *Nominé pour le prix du meilleur article*.

Brevet

Brevet français déposé le 12 décembre 2007 sous le numéro FR 0759765 intitulé : “Cartes auto-organisatrices à liens topologiques valués, procédé de commande d'un système d'apprentissage automatique utilisant une telle carte auto-organisatrice et analyseur de comportements dynamiques”. Ce brevet a fait l'objet d'une extension PCT demande internationale n° PCT/FR2008/052288 régularisé par une entrée en phase nationale aux Etats-Unis sous le n°12/747855, au Canada (n°dépôt à venir) et en phase régionale européenne sous le n° 08865257.3.

Protection Logiciel

Dépôt de logiciel intitulé “Density-based Simultaneous 2-Level – Self Organizing Map (DS2L-SOM)” sous le n° IDDN.FR.001.490019.000S.P.2008.000.20000 auprès de l'Agence pour la Protection des Programmes.

Bibliographie

[1] C. Aggarwal and P. Yu. A Survey of Synopsis Construction Methods in Data Streams. In C. Aggarwal, editor, *Data Streams : Models and Algorithms*, pages 169–207. Springer, 2007.

[2] C. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *Special Interest Group on Management of Data Conference*, pages 575–586, 2003.

[3] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Very Large Data Base*, pages 81–92, 2003.

[4] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan. Dynamic Self Organising Maps with Controlled Growth for Knowledge Discovery. *IEEE Transactions on Neural Networks*, 11(3) :601–614, 2000.

[5] R. Alhajj, O. Abul, and F. Polat. Cluster stability analysis using sub-sampling, 2004. unknown publisher.

[6] M. Aupetit. Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing*, 70(7-9) :1304–1330, 2007.

[7] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *ACM SIAM Symposium on Discrete Algorithms*, pages 633–634, 2002.

[8] A. Balzanella, Y. Lechevallier, and R. Verde. A new approach for clustering multiple streams of data. In S. Ingrassia and R. Rocci, editors, *Classification and Data Analysis*, pages 417–420, 2009.

[9] D. Beaton and I. Valora. RADDACL : A recursive algorithm for clustering and density discovery on non-linearly separable data. *Prodeeding of International Joint Conference on Neural Networks*, pages 1423–1428, Aug. 2007.

[10] S. Ben-David and U. von Luxburg. Relating clustering stability to properties of cluster boundaries. In *COLT*, pages 379–390, 2008.

[11] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4) :281–297, 1999.

[12] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In *Pacific Symposium on Biocomputing*, volume 7, pages 6–17, 2002.

[13] A. Ben-Israel and T. N. E. Greville. *Generalized Inverse : Theory and Applications.* Springer Verlag, New York, 2003.

[14] Y. Bennani. *Apprentissage Connexionniste.* Editions Hermès Science, 2006.

[15] A. Bertoni and G. Valentini. Model order selection for bio-molecular data clustering. *BMC Bioinformatics*, 8(S-2), 2007.

[16] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM : the generative topographic mapping. *Neural Computation*, 10 :215–234, 1998.

[17] M. Bittner, P. Meltzer, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dor, N. Sampas, E. Dougherty, E. Wang, F. Marincola, C. Gooden, J. Lueders, A. Glatfelter, P. Pollock, J. Carpten, E. Gillanders, D. Leja, K. Dietrich, C. Beaudry, M. Berens, D. Alberts, V. Sondak, N. Hayward, and J. Trent. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, 406(6795) :536–540, 2000.

[18] H.-H. Bock and E. Diday. *Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data.* Springer Verlag, Heidelberg, 2000.

[19] E. L. J. Bohez. Two level cluster analysis based on fractal dimension and iterated-function systems (ifs) for speech signal recognition. *IEEE Asia-Pacific Conference on Circuits and Systems*, pages 291–294, 1998.

[20] D. Bouchaffra and J. Tan. Structural hidden markov models using a relation of equivalence : Application to automotive designs. *Data Mining and Knowledge Discovery*, 12 :79 – 96, 2006.

[21] R. Boulet, B. Jouve, F. Rossi, and N. Villa. Batch kernel som and related laplacian methods for social network analysis. *Neurocomputing*, 71(7-9) :1257–1273, 2008.

[22] A. Bowman. An Alternative Method of Cross-Validation for the Smoothing of Density Estimates. *Biometrika*, 71 :353–360, 1984.

[23] U. Braun, C. Peeters, and B. Hölldobler. The Giant Nests of the African Stink Ant Paltothyreus tarsatus (Formicidae, Ponerinae). *Biotropica*, 26(3) :308–311, 1994.

[24] J. Bruske and G. Sommer. Dynamic cell structures. In *Advances in Neural Information Processing Systems (NIPS'94)*, pages 497 – 504, 1994.

[25] T. Calinski and J. Harabasz. Dendrite method for cluster analysis. *Communications in Statistics*, 3(1) :1–27, 1974.

[26] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *2006 SIAM Conference on Data Mining*, pages 328–339, 2006.

[27] M. Chavent, F. De Carvalho, Y. Lechevallier, and R. Verde. New clustering methods for interval data. *Computational Statistics*, 21(23) :211–230, 2006.

[28] Y. Cheung and L. Law. Rival-Model Penalized Self-Organizing Map. *IEEE Trans. Neural Networks*, 18(1) :289–295, 2007.

[29] T. W. S. Chow and S. Wu. An online cellular probabilistic self-organizing map for static and dynamic data sets. *IEEE Transactions on Circuit and System*, 51(4) :732–747, 2004.

[30] G. Cleuziou, L. Martin, V. Clavier, and C. Vrain. DDOC : overlapping clustering of words for document classification. In *SPIRE*, pages 127–128, 2004.

[31] B. Conan-Guez and F. Rossi. Speeding up the dissimilarity self-organizing maps by branch and bound. In *IWANN*, pages 203–210, 2007.

[32] B. Conan-Guez, F. Rossi, and A. El Golli. Fast algorithm and implementation of dissimilarity self-organizing maps. *Neural Networks*, 19(6–7) :855–863, Aug. 2006.

[33] A. Cornuéjols, L. Miclet, and Y. Kodratoff. *Apprentissage Artificiel : Concepts et algorithmes*. Eyrolles, 2002.

[34] C. Cottrell and P. Letremy. Working times in atypical forms of employment : the special case of part-time work. In C. Lesage and M. Cottrell, editors, *Conf. ACSEG, Rennes ; Connexionnist Approaches in Economics and ManagementSciences*. Kluwer, 2001.

[35] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 1(2) :224–227, 1979.

[36] A. Dejean, G. Beugnon, and J.-P. Lachaud. Spatial components of foraging behaviour in an African ponerine ant, Paltothyreus tarsatus. *Journal of Insect Behaviour*, 6 :271–285, 1993.

[37] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39 :1–38, 1977.

[38] J.-L. Deneubourg, S. Goss, J.-M. Pasteels, D. Fresneau, and J.-P. Lachaud. *From Individual Characteristics to Collective Organisation : the example of Social Insects*, volume 54, chapter Self-organization mechanisms in ant societies (II) : learning in foraging and division of labor, pages 197–217. Experientia suppl. Birkhäuser Verlag, Basel., 1987.

[39] S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7) :1–21, 2002.

[40] J. C. Dunn. Well separated clusters and optimal fuzzy partitions. *J. Cybern.*, 4 :95–104, 1974.

[41] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.

[42] A. Dussutour, V. Fourcassie, D. Helbing, and J.-L. Deneubourg. Optimal traffic organization in ants under crowded conditions. *Nature*, 428(6978) :70–3, 2004.

[43] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, pages 147–153, 2003.

[44] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[45] W. Fan, Y. an Huang, H. Wang, and P. S. Yu. Active mining of data streams. In M. W. Berry, U. Dayal, C. Kamath, and D. B. Skillicorn, editors, *SDM*. SIAM, 2004.

[46] G. Frahling and C. Sohler. A fast k-means implementation using coresets. In *SCG '06 : Proceedings of the twenty-second annual symposium on Computational geometry*, pages 135–143, New York, NY, USA, 2006. ACM.

[47] C. Fraley and A. Raftery. How many clusters ? Which clustering methods ? Answers via model-based cluster analysis. *Computer Journal*, 41 :578–588, 1998.

[48] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *J. Am. Stat. Assoc.*, 97(458) :611–631, 2002.

[49] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[50] D. Fresneau. *Biologie et comportement social d'une fourmis ponérine néotropicale (Pachycondyla apicalis).* PhD thesis, Université Paris-Nord (Paris 13), Paris, 1994.

[51] D. Fresneau, B. Corbara, and J. Lachaud. Organisation Sociale et Structuration Spatiale Autour du Couvain chez Pachycondyla apicalis. *Actes coll. Insectes Sociaux*, 5 :83–92, 1989.

[52] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315 :972–976, 2007.

[53] J. Fridlyand and S. Dudoit. Applications of resampling methods to estimate the number of clusters and to improve the accuracy of a clustering method. Technical Report 600, Stat. Berkeley, 2001.

[54] B. Fritzke. Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9) :1441–1460, 1994.

[55] B. Fritzke. Growing grid - a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5) :9–13, 1995.

[56] D. P. Fromberg and D. Bergen. *Play from birth to Twelve : Contexts, perspectives, and Meanings.* Routledge, New York, 2006.

[57] U. Galassi, A. Giordana, and L. Saitta. Incremental construction of structured hidden markov models. In M. M. Veloso, editor, *IJCAI*, pages 798–803, 2007.

[58] V. Ganti, J. Gehrke, R. Ramakrishnan, and W.-Y. Loh. A framework for measuring differences in data characteristics. *J. Comput. Syst. Sci.*, 64(3) :542–578, 2002.

[59] J. Gehrke, F. Korn, and D. Srivastava. On computing correlated aggregates over continual data streams. In *Special Interest Group on Management of Data Conference*, pages 13–24, 2001.

[60] S. Guérif and Y. Bennani. Selection of clusters number and features subset during a two-levels clustering task. In *Proceeding of the 10th International Conference Artificial intelligence and Soft Computing 2006*, pages 28–33, Palma de Mallorca, Spain, August 2006.

[61] S. Guha and B. Harb. Approximation algorithms for wavelet transform coding of data streams. *IEEE Transactions on Information Theory*, 54(2) :811–830, 2008.

[62] S. Guha, R. Rastogi, and K. Shim. Cure : an efficient clustering algorithm for large databases. *Special Interest Group on Management of Data Record*, 27(2) :73–84, 1998.

[63] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On Clustering Validation Techniques. *Journal of Intelligent Information Systems*, 17(2-3) :107–145, 2001.

[64] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Cluster Validity Methods. *Special Interest Group on Management of Data Record*, 31(2,3) :40–45, 19–27, 2002.

[65] J. Handl and J. Knowles. Multiobjective clustering with automatic determination of the number of clusters. Technical report, UMIST, Manchester, UK, 2004.

[66] A. Hardy and N. Kasoro Mulenda. Une nouvelle méthode de classification pour des données intervalles. *Mathématiques et Sciences Humaines*, 187 :79–91, 2009.

[67] J. A. Hartigan and M. A. Wong. Algorithm AS 136 : A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 28(1) :100–108, 1979.

[68] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76 :175–181, 1999.

[69] J. R. Hershey and P. A. Olsen. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 317–320, 2007.

[70] B. Hölldobler. Canopy orientation : a new kind of orientation in ants. *Science*, 210 :86–88, 1980.

[71] B. Hölldobler. Communication during foraging and nest-relocation in the African stink ant, Paltothyreus tarsatus. *Z.Tierpsychol*, 65 :40–52, 1984.

[72] B. Holldobler and E. Wilson. *The ants.* Harvard University Press, Cambridge, MA, 1990.

[73] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, San Francisco, CA, 2001. ACM Press.

[74] M. F. Hussin, M. S. Kamel, and M. H. Nagi. An efficient two-level SOMART document clustering through dimensionality reduction. In *ICONIP*, pages 158–165, 2004.

[75] S. Ibbou. *Classification, analyse des correspondances et méthodes neuronales.* PhD thesis, Université Paris 1, Paris, 1998.

[76] A. Jain and P. Flynn. Image segmentation using clustering. In *Advances in Image Understanding : A Festschrift for Azriel Rosenfeld*, 65–83. IEEE Press, 1996.

[77] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[78] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering : a review. *ACM Computing Surveys*, 31(3) :264–323, 1999.

[79] D. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1) :1–13, 1977.

[80] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm : Analysis and implementation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24 :881–892, 2002.

[81] G. Karypis, E.-H. Han, and V. Kumar. Chameleon : Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8) :68–75, 1999.

[82] G. Karypis and V. Kumar. hMETIS 1.5 : A Hypergraph Partitioning Package. Technical report, Dept. of Computer Science, Univ. of Minnesota, 1998.

[83] R. Kassab and F. Alexandre. Incremental data-driven learning of a novelty detection model for one-class classification with application to high-dimensional noisy data. *Machine Learning*, 2007.

[84] M. K. Kerr and G. A. Churchill. Bootstrapping cluster analysis : Assessing the reliability of conclusions from microarray experiments. *PNAS*, 98 :8961–8965, 2000.

[85] K. Kiviluoto. Topology Preservation in Self-Organizing Maps. *International Conference on Neural Networks*, pages 294–299, 1996.

[86] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 1984.

[87] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 2001.

[88] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. SOM PAK : The self-organizing map program package. Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland., 1996.

[89] E. E. Korkmaz. A two-level clustering method using linear linkage encoding. In *International Conference on Parallel Problem Solving From Nature, Lecture Notes in Computer Science*, volume 4193, pages 681–690. Springer-Verlag, 2006.

[90] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

[91] T. Lange, V. Roth, M. L. Braun, and J. M. Buhmann. Stability-based validation of clustering solutions. *Neural Comput.*, 16(6) :1299–1323, 2004.

[92] J. S. Larson, E. Bradlow, and P. Fader. An Exploratory Look at Supermarket Shopping Paths. *International Journal of Research in Marketing*, 2005.

[93] L. Lebart, A. Morineau, and M. Piron. *Statistique exploratoire multidimensionnelle*. Dunod, 1995.

[94] H. Lee and S. Cho. Som-based novelty detection using novel data. In M. Gallagher, J. M. Hogan, and F. Maire, editors, *IDEAL*, volume 3578 of *Lecture Notes in Computer Science*, pages 359–366. Springer, 2005.

[95] E. Levine and E. Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13(11) :2573–2593, 2001.

[96] P. Lyman and H. R. Varian. How Much Information, 2003. Retrieved from http ://www.sims.berkeley.edu/how-much-info-2003.

[97] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Very Large Data Base*, pages 346–357, 2002.

[98] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *Special Interest Group on Management of Data Conference*, pages 251–262, 1999.

[99] J. Marron and M. Wand. Exact mean integrated squared error. *The Annals of Statistics*, 20 :712–736, 1992.

[100] S. Marsland, J. Shapiro, and U. Nehmzow. A self-organising network that grows when required. *Neural Networks*, 15(8-9) :1041–1058, 2002. [There is a reference to a similar piece of work omitted from this paper. It is : A. Baraldi and P. Blonda, 'A survey of fuzzy clustering algorithms for pattern recognition : Part II', IEEE Trans. Systems, Man and Cybernetics - Part B : Cybernetics, vol. 29, no. 6, pp. 786-801, Dec. 1999.].

[101] T. Martinetz. Competitive Hebbian Learning rule forms perfectly topology preserving maps. In S. Gielen and B. Kappen, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN-93), Amsterdam*, pages 427–434, Heidelberg, 1993. Springer.

[102] T. M. Martinetz and K. J. Schulten. A "neural-gas" network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402. Elsevier Science Publishers, Amsterdam, 1991.

[103] H. Matsushita and Y. Nishio. Self-Organizing Map with False-Neighbor Degree between Neurons for Effective Self-Organization. *IEICE Transactions on Fundamentals*, E91-A(6) :1463–1469, 2008.

[104] D. W. Matula. Graph Theoretic Techniques for Cluster Analysis Algorithms. In J. V. Ryzin, editor, *Classification and Clustering*, pages 95–129. Academic Press, New York, 1977.

[105] G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley and Sons, New York, 2000.

[106] J. B. McQueen. Some methods of classification and analysis of multivariate observations. In L. L. Cam and J. Neyman, editors, *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, University of California Press, 1967.

[107] M. Meila. Comparing clusterings by the variation of information. In *COLT*, pages 173–187, 2003.

[108] G. Milligan and M. Cooper. An Examination of Procedures for Determining the Number of Clusters in a Data Set. *Psychometrika*, 50 :159–179, 1985.

[109] U. Möller and D. Radke. A cluster validity approach based on nearest-neighbor resampling. In *ICPR (1)*, pages 892–895, 2006.

[110] T. Monnin and C. Peeters. Monogyny and regulation of worker mating in the queenless ant Dinoponera quadriceps. *Animal. Behavior*, 55 :299–306, 1998.

[111] K. Murphy. *Dynamic Bayesian Networks : Representation, Inference and Learning*. Ph.D thesis, Dpt. of Computer Science, UC, Berkeley, 2002.

[112] P. Natarajan and R. Nevatia. View and scale invariant action recognition using multiview shape-flow models. In *proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008*, pages 1–8, 2008.

[113] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering : Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.

[114] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. In *Proceedings of IEEE International Conference on Data Engineering*, 2002.

[115] A. Ocsa, C. Bedregal, and E. Cuadros-Vargas. DB-GNG : A constructive self-organizing map based on density. *Prodeeding of International Joint Conference on Neural Networks*, pages 1506–1511, August 2007.

[116] S. R. Pamudurthy, S. Chandrakala, and C. C. Sakhar. Local density estimation based clustering. *Prodeeding of International Joint Conference on Neural Networks*, pages 1338–1343, August 2007.

[117] H. Park and T. Ozeki. Singularity and Slow Convergence of the EM algorithm for Gaussian Mixtures. *Neural Process Letters*, 29 :45–59, 2009.

[118] C. Peeters. Monogyny and polygyny in ponerine ants with or without queens. In L. Keller, editor, *Queen Number and Sociality in Insects*, pages 235–261. Oxford University Press, Oxford, 1993.

[119] C. Peeters. Morphologically "primitive" ants : comparative review of social characters, and the importance of queen-worker dimorphism. In J. Choe and B. Crespi, editors, *The Evolution of Social Behaviour in Insects and Arachnids*. Cambridge, Cambridge, 1997.

[120] A. Pezon, D. Denis, P. Cerdan, J. Valenzuela, and D. Fresneau. Queen movement during colony emigration in the facultatively polygynous ant Pachycondyla obscuricornis. *Naturwissenschaften*, 92 :35–39, 2005.

[121] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2) :257–286, 1989.

[122] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NY, 1993.

[123] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336) :846–850, Dec. 1971.

[124] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2) :195–239, 1984.

[125] E. Robinson, T. Richardson, A. Sendova-Franks, O. Feinerman, and N. Franks. Radio tagging reveals the roles of corpulence, experience and social information in ant decision making. *Behavioral Ecology and Sociobiology*, 2009.

[126] V. Roth, M. Braun, T. Lange, and J. Buhmann. A resampling approach to cluster validation. *In Computational Statistics - COMPSTAT'02*, 2002. To appear.

[127] M. Rudemo. Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, 9 :65–78, 1982.

[128] S. Sain, K. Baggerly, and D. Scott. Cross-Validation of Multivariate Densities. *Journal of the American Statistical Association*, 89 :807–817, 1994.

[129] J. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computer*, 18(5) :401–409, May 1969.

[130] E. H. Sbai. La classification automatique par les statistiques d'ordre. *Traitement du signal*, 16(6) :437–449, 1999.

[131] R. Schweller, A. Gupta, E. Parsons, and Y. Chen. Reversible sketches for efficient and accurate change detection over network data streams. In *IMC'04 : proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 207–212, New York, NY, USA, 2004.

[132] D. W. Scott. *Multivariate Density Estimation : Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley-Interscience, September 1992.

[133] D. W. Scott and S. R. Sain. *multi-dimensional density estimation*, pages 229–263. Elsevier, Amsterdam, 2004.

[134] G. Sempo, S. Canonge, C. Detrain, and J.-L. Deneubourg. Complex Dynamics Based on a Quorum : Decision-Making Process by Cockroaches in a Patchy Environment. *Ethology*, 115 :1–12, 2009.

[135] O. Shamir and N. Tishby. Cluster Stability for Finite Samples. In *Twenty-First Annual Conference on Neural Information Processing Systems (NIPS 2007)*, 2007.

[136] S. Sheather and M. Jones. A Reliable Data-Based Bandwidth Selection Method For Kernel Density Estimation. *Journal of the Royal Statistical Society, Series B*, 53 :683–690, 1991.

[137] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8) :888–905, 2000.

[138] B. Silverman. Using kernel density estimates to investigate multi-modality. *Journal of the Royal Statistical Society, Series B*, 43 :97–99, 1981.

[139] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, April 1986.

[140] T. Smith and D. Alahakoon. Growing self-organizing map for online continuous clustering. In *Foundations of Computational Intelligence (4)*, pages 49–83. 2009.

[141] M. Smolkin and D. Ghosh. Cluster stability scores for microarray data in cancer studies. *BMC Bioinformatics*, 4 :36, 2003.

[142] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy : The Principles and Practice of Numerical Classification*. W.H. Freeman and Company, San Francisco, USA, 1973.

[143] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In M. Grobelnik, D. Mladenic, and N. Milic-Frayling, editors, *KDD-2000 Workshop on Text Mining, August 20*, pages 109–111, Boston, MA, 2000.

[144] K. Taşdemir and E. Merényi. Data topology visualisation for the self-organizing map. *European Symposium on Artificial Neural Networks (ESANN'2006), Bruges, Belgium, d-side eds.*, pages 277–282, Apr. 2006.

[145] K. Taşdemir and E. Merényi. A new cluster validity index for prototype based clustering algorithms based on inter- and intra-cluster density. In *Prodeeding of International Joint Conference on Neural Networks*, pages 1560–1566, August 2007.

[146] S. Theodoridis and K. Koutroumbas. *Pattern Recognition 3rd ed*, page 635. 2006.

[147] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic, 2000.

[148] E. Tsivtsivadze, T. Pahikkala, J. Boberg, and T. Salakoski. Locality-convolution kernel and its application to dependency parse ranking. In M. Ali and R. Dapoigny, editors, *IEA/AIE*, volume 4031 of *Lecture Notes in Computer Science*, pages 610–618. Springer, 2006.

[149] A. Ultsch. Clustering with SOM : U*C. In *Procedings of the Workshop on Self-Organizing Maps*, pages 75–82, 2005.

[150] J. J. Verbeek, N. Vlassis, and B. Kröse. The generative self-organizing map : a probabilistic generalization of kohonen's som. Technical report, 2002.

[151] J. Vesanto. Neural network tool for data mining : SOM Toolbox, 2000.

[152] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. Self-Organizing Map in Matlab : the SOM Toolbox. *Proceedings of the Matlab DSP Conference*, pages 35–40, 1999.

[153] T. Villmann and H.-U. Bauer. Applications of the growing self-organising map. *Neurocomputing*, 21 :91 – 100, 1998.

[154] L. Vincent and P. Soille. Watersheds in digital spaces : An efficient algorithm based on immersion simulation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13 :583–598, 1991.

[155] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4) :395–416, 2007.

[156] U. von Luxburg. A high-level summary of theoretical results on clustering stability. *Submitted*, 2009.

[157] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301) :236–244, 1963.

[158] Z. Wu and R. Leahy. An approximation method of evaluating the joint likelihood for first-order GMRFs. *IEEE Transactions on Image Processing*, 2(4) :520–523, 1993.

[159] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *ICDE '98 : Proceedings of the Fourteenth International Conference on Data Engineering*, pages 324–331, Washington, DC, USA, 1998. IEEE Computer Society.

[160] H. Yin. On the equivalence between kernel self-organising maps and self-organising mixture density networks. *Neural Networks*, 19(6-7) :780–784, 2006.

[161] S.-H. Yue, P. Li, J.-D. Guo, and S.-G. Zhou. Using greedy algorithm : DBSCAN revisited II. *Journal of Zhejiang University SCIENCE*, 5(11) :1405–1412, 2004.

[162] S.-H. Yue, P. Li, J.-D. Guo, and S.-G. Zhou. A statistical information-based clustering approach in distance space. *Journal of Zhejiang University SCIENCE*, 6(1) :71–78, 2005.